applied sciences

MDPI

Article

# DUKMSVM: A Framework of Deep Uniform Kernel Mapping Support Vector Machine for Short Text Classification

**Zhaoying Liu [1], Haipeng Kan [1], Ting Zhang [1,*] and Yujian Li [2]**

[1] Faculty of Information Technology, Beijing University of Technology, Beijing 100124, China; zhaoying.liu@bjut.edu.cn (Z.L.); lzying219@163.com (H.K.)

[2] School of Artificial Intelligence, Guilin University of Electronic Technology, Guilin 541004, China; liyujian@guet.edu.cn

* Correspondence: zhangting@bjut.edu.cn

Received: 19 January 2020; Accepted: 26 March 2020; Published: 30 March 2020


**Abstract:** This paper mainly deals with the problem of short text classification. There are two main contributions. Firstly, we introduce a framework of deep uniform kernel mapping support vector machine (DUKMSVM). The significant merit of this framework is that by expressing the kernel mapping function explicitly with a deep neural network, it is in essence an explicit kernel mapping instead of the traditional kernel function, and it allows better flexibility in dealing with various applications by applying different neural network structures. Secondly, to validate the effectiveness of this framework and to improve the performance of short text classification, we explicitly express the kernel mapping using bidirectional recurrent neural network (BRNN), and propose a deep bidirectional recurrent kernel mapping support vector machine (DRKMSVM) for short text classification. Experimental results on five public short text classification datasets indicate that in terms of classification accuracy, precision, recall rate and F1-score, the DRKMSVM achieves the best performance with the average values of accuracy, precision, recall rate, and F1-score of 87.23%, 86.99%, 86.13% and 86.51% respectively compared to traditional SVM, convolutional neural network (CNN), Naive Bayes (NB), and Deep Neural Mapping Support Vector Machine (DNMSVM) which applies multi-layer perceptron for kernel mapping.

**Keywords:** short text classification; support vector machine; recurrent neural network; kernel mapping

## 1. Introduction

With the rapid increase of communication on the internet, a large amount of textual data has been generated. How to mine important and useful information from these textual data is of great significance to promote the development of various industries [1,2]. Text classification is one of the important information retrieval and data mining technologies, the purpose of text classification is to divide a given text into one or multiple predefined categories according to the extracted features [3]. It is an important research direction in natural language processing, and has been widely used in various applications, such as information retrieval, document classification, sentiment analysis and so on [4]. Among them, short text classification has attracted more attention and encountered more challenges due to their limited length. Most of the internet data are short texts, such as microblogs and bulletin board system (BBS) [5]. Accurate short text classification plays an important role both for the enterprises and individuals, even for the government services. It can help the enterprises to understand public preferences in a better way, thus to quickly grasp business opportunities, and increase revenue. For the users, it is also helpful for solving the problem of network information chaos, and facilitate users to hunt information and get the useful information needed. For the government services, it assists

the government to understand public opinions and provide better services [6]. With the explosively increasing short texts on the Internet, how to efficiently generate practical value from these massive short texts is an urgent problem to be solved [7].

In the past decades, scholars have made various attempts to improve the accuracy of short text classification [8]. Many efforts have been made in two aspects: text representation and classifier. Text representation is the basis of text classification system, and its modeling quality determines the performance of the classification system. Since the traditional text representation methods model words according to the word frequency, when applied to represent words characterized by high dimension and high sparsity, it would result in two difficulties: one is that the semantic features extracted are insufficient; the other is that it is difficult to support the short text classification task with increasing data volume since it mainly depends on human beings to extract features [9]. Currently, the distributed representation of words is a popular method in the field of text presentation. Its principle is to map words to fixed dense data vectors, namely Word Embedding [10]. Compared with traditional one-hot Encoding [11,12], bag-of-word model [13], and vector space model [14,15], etc., the distributed representation possesses relatively good semantic feature expression competence, and in the meantime data form can be read and processed efficiently by neural networks. In recent years, word vector is widely popular in the field of text semantic modeling, which can be attributed to Google's open-source Word2vec word vector tool [16]. Upon its launch, research on text classification model has been transformed from the conventional shallow machine learning models to the deep learning models. By means of vector in mathematics, words are able to map the text input to a low dimensional, continuous and dense vector matrix, so that deep neural networks can be transferred to text categorization, which greatly promotes the development of related researches

In terms of classifiers, there are also various methods that have been put forward, such as NB [17–19], K-Nearest Neighbor (KNN) [20–22], SVM [23] and so on. For instance, Han et al. improved the performance of NB in terms of text classification by using two heuristic methods: text normalization and feature weights [20]. Guo et al. [21] proposed a KNN model-based classifier named KNN model, which combines the advantages of both KNN and Rocchio. Zhao et al. [19] optimized and improved the fundamental formula in NB algorithm and come up with CIT-NB algorithm, which combined the core ideas of category and the improved Term Frequency-Inverse Document Frequency (TF-IDF) algorithm. Dadgar et al. [23] proposed a SVM method based on semi-supervised learning for short text classification. By using TF-IDF as the weighting function, and using semi-supervised learning algorithm for iterative training, this method can improve the performance of traditional SVM method. Among these methods, SVM achieves better results than other models in the light of short text classification [11,24]. With the help of kernel learning, the classification performance of SVM can be further improved [10]. However, the selection of kernel functions has great influence on the classification performance. Therefore, different kernel functions may lead to different classification results, and a single kernel function is not applicable to all types of short text classification. Relevant studies have shown that the compound kernel generated by multi-kernel learning is not always superior to the base kernel, which may attribute to shallow network structure of multi-kernel learning. In other words, multi-kernel learning has limitations to some extent, and is not competent enough to express relatively complicated compound kernel via linear or nonlinear combination of the base kernels [25].

Recently, models based on deep learning have attracted more attention [26]. Deep learning is an important branch in the field of machine learning.The concept of deep learning was first proposed by Hinton et al., whose research on artificial neural network was published in *Science* in 2006 [14]. Now, as a prevalent network at present, the convolutional neural network has attained satisfying results in many fields due to its powerful ability of feature extraction [27,28]. In the light of text classification, Kim applied the convolutional neural network to classify short film criticism [16]. Meng et al. introduced the attention mechanism to CNN and successfully applied it to document modeling [29]. Besides the multi-layer perceptron and CNN, another neural network structure with recurrent unit named

Recurrent Neural Network (RNN) has attracted more attention on text classification [30,31]. In fact, a piece of short text is actually a short sequence in which the occurrence of each word is correlated with its context. RNN is a kind of neural networks which considering both the current input and the previous information for generating the current output. Therefore, comparing with multi-layer perceptron and CNN, RNN is more popular for processing sequence data. However, one shortcoming of the conventional RNN is that they just make use of the previous context, the future context is still not exploited [32]. To overcome the limitations, Schuster M. et. al [33] proposed a bidirectional recurrent neural network (BRNN). Given a specific time frame, BRNN can take into account all available input information in the past and future, therefore it can effectively retain the context features of the sequence data, such as text data.

To solve the kernel learning problem existing in SVM, and considering the mapping ability of deep neural networks, Li et al. [34] introduce a short text classification method based on DNMSVM. It is able to explicitly express the kernel with a multilayer perceptron, and map the original text data to an appropriate dimensional space. Meanwhile, SVM is applied for classification in that appropriately mapped dimensional space. Theoretically, DNMSVM can approximate any kernel function. However, as mentioned above, there are various neural network structures besides multi-layer perceptron, and the BRNN has more advantages on sequence data processing. Therefore, motivated by DNMSVM and considering the advantages of BRNN, we extended DNMSVM to a framework of deep uniform kernel mapping support vector machine (D*U*KMSVM), and propose a short text classification based on deep recurrent kernel mapping support vector machine (DRKMSVM). Different from the DNMSVM, various neural network structures can be used for kernel mapping in D*U*KMSVM. To improve the performance of short text classification, we express the kernel mapping function with BRNN, thus to map the originally high-dimension textual features to an appropriate feature space. Then, in the mapped feature spaces, we apply the SVM for the final classification. Combining the powerful sequence data processing ability of BRNN helps to improve the performance of short text classification.

The rest of this paper is organized as follows: Section 2 introduces the framework of D*U*KMSVM briefly; Section 3 describes the DRKMSVM model and its learning algorithm in details; Section 4 demonstrates the main experimental results and analysis; finally, Section 5 gives conclusions.

## 2. The Framework of Deep Uniform Kernel Mapping Support Vector Machine (D*U*KMSVM)

SVM is a binary classification model with the learning objective to find the hyper-plane with maximum margin in the feature space so as to divide data into positive or negative classes [35]. It is well known that with the help of kernel learning, the classification performance of SVM can be further improved [10]. Given that a problem is linearly inseparable in the original space, $\boldsymbol{x}^l \in R^r$ will be mapped to $\Phi(\boldsymbol{x}^l) \in H^{r'}$ by adopting the nonlinear mapping $\Phi : R^r \longrightarrow H^{r'} (r' > r)$, and then the theory and method in the case of linear separability are used to solve the problem. Herein, $\Phi$ is called the kernel mapping, and is normally an unknown function. Notably, it can be implicitly induced by the kernel function $K(\boldsymbol{x}, \boldsymbol{x}') = (\boldsymbol{x} \cdot \boldsymbol{x}' + 1)^q$ , where "$\cdot$" refers to inner product. The three commonly used kernel functions are linear kernel $K(\boldsymbol{x}, \boldsymbol{x}') = \boldsymbol{x}^T \cdot \boldsymbol{x}'$, polynomial kernel $K(\boldsymbol{x}, \boldsymbol{x}') = (1 + \boldsymbol{x}^T \cdot \boldsymbol{x}')^q$, and radial basis function $K(\boldsymbol{x}, \boldsymbol{x}') = exp(-\gamma \| \boldsymbol{x} - \boldsymbol{x}' |^2)$.

However, as mentioned above, the selection of kernel functions has great influence on the classification performance. A single kernel function is not applicable to all types of short text classification, while multi-kernel learning is also not competent enough to express relatively complicated compound kernel via linear or nonlinear combination of the base kernels. Therefore, to solve the kernel learning problem existing in SVM, Li et al. [34] introduced a DNMSVM. It explicitly expresses the kernel with a multi-layer perceptron, refer to [34] for more details. Motivated by DNMSVM, we present the framework of D*U*KMSVM, where *U* stands for different deep neural network, as illustrated in Figure 1.

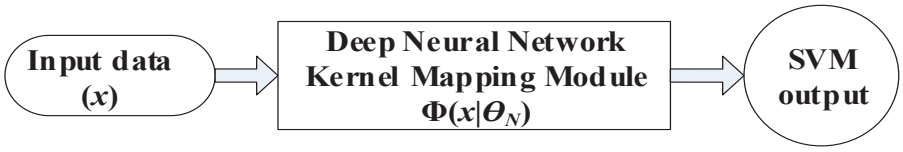



**Figure 1.** The architecture of deep uniform kernel mapping support vector machine (D*U*KMSVM).

To explain this framework more clearly, we take multi-layer perceptron kernel mapping (DMKMSVM) as an example, as shown in Figure 2. It is equivalent to the DNMSVM model. The essence of DMKMSVM is to express the mapping kernel explicitly with a multi-layer perceptron, and then taking the output of multi-layer perceptron as the input of linear SVM, it will output the text classification results.

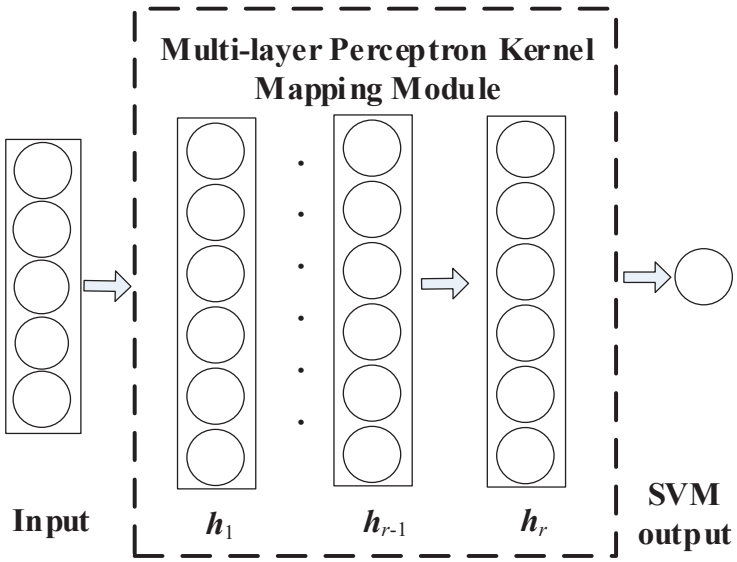



**Figure 2.** The architecture of multi-layer perceptron kernel mapping (DMKMSVM).

It can be seen from Figure 2 that DMKMSVM contains two modules, namely feature extraction module and classification module. The feature extraction module defines a kernel mapping, which first represents the input data with TF-IDF, and then maps to implicit vectors using neural network, while the classification module classifies the last implicit vector using SVM. Suppose that $\boldsymbol{W}^j$ stands for the weight matrix between the hidden layers $\boldsymbol{h}_{j-1}$ and $\boldsymbol{h}_j$; $\boldsymbol{b}^j$ refers to the bias vector of the layer $\boldsymbol{h}_j$; $\boldsymbol{W}^{r+1}$ and $b^{r+1}$ are the weight and bias of the output layer, respectively. For the $l$-th sample ($l$ is a positive integer), the calculation process of each layer of DMKMSVM is as follows:

$$\begin{cases} \boldsymbol{h}_0^l = \boldsymbol{x}^l, \\ \boldsymbol{h}_j^l = \sigma(\boldsymbol{W}^j \boldsymbol{h}_{j-1}^l + \boldsymbol{b}^j), 1 \le j \le r-1 \\ \phi(\boldsymbol{x}^l) = \sigma\left(\boldsymbol{W}^r \boldsymbol{h}_{r-1}^l + b^r\right), \\ o^l = \boldsymbol{W}^{r+1} \phi(\boldsymbol{x}^l) + b^{r+1}, \end{cases} \tag{1}$$

where $\sigma(\cdot)$ is the nonlinear activation function. As a matter of fact, given the input $\boldsymbol{x}^l$, $\phi\left(\boldsymbol{x}^l\right)$ is a function defined by multilayer perceptron. Specifically, it will map the input to $\phi\left(\boldsymbol{x}^l\right)$ through multiple hidden layers $\boldsymbol{h}_1^l, \boldsymbol{h}_2^l, \cdots, \boldsymbol{h}_{r-1}^l$. In short, DMKMSVM is a SVM concerning $\phi\left(\boldsymbol{x}^l\right)$, and its output is the classification result $\phi\left(\boldsymbol{x}^l\right)$ utilizing SVM as a classifier.

In the course of training, the objective function of DMKMSVM is:

$$\begin{aligned} L_N = & \min_{\substack{W^j, b^j, \xi_l \\ 1 \le j \le r+1, 1 \le l \le N}} \frac{1}{2}||W^{r+1}||^2 + C\sum_{l=1}^{N}\xi_l^2 \\ & s.t. y^l(W^{r+1}\phi(x^l) + b^{r+1}) \ge 1 - \xi_l, \xi_l \ge 0, l = 1, ..., N \end{aligned} \tag{2}$$

where $W^{r+1}$ and $b^{r+1}$ refer to the weight and bias of the SVM layer, respectively. Hence, the issue of function constraint extremum can be equivalently converted to the following optimization problem without constraint:

$$L_N = \arg\min_{\theta} \frac{1}{2}||W^{r+1}||^2 + C\sum_{l=1}^{N}\left[max(1 - y^l \times (W^{r+1}\phi(x^l|\theta) + b^{r+1}), 0)\right]^2 \tag{3}$$

where $\theta = \left\{W^j, b^j, 1 \le j \le r\right\} \cup \left\{W^{r+1}, b^{r+1}\right\}$ are all the parameters to be learned, and $y^l$ is the label of $x^l$, whose value is $+1$ or $-1$.

In order to train DMKMSVM effectively, we take advantage of the two-stage training method integrating unsupervised pre-training with Restricted Boltzman Machine (RBM) and supervised fine-tuning. The detailed description is illustrated in Algorithm 1.

**Algorithm 1** Joint training of DMKMSVM

**Input:**Training set $S = \{(x^l, y^l), 1 \le l \le N\}$, network architecture, max number of epoch
**Output:** $\theta = \{W^j, b^j, 1 \le j \le r+1\}$
**Stage One: Pre-training**
1. Randomly initialize $W^j \approx 0$ ,$b^j \approx 0$ , $1 \le j \le r+1$ ;
2. Learn $W^j$ and $b^j$ for every $RBM$ of $h_{j-1}$ and $h_j$ ,$1 \le j \le r$ ;
**Stage Two: Fine-tuning**
3. **For** $epoch = 1$ to $maxepoch$ do
4. **For** $l$=1 to $N$ do
5. Compute $h_0 = x^l, u_j^l = W^j h_{j-1} + b^j, h_j = \sigma\left(u_j^l\right), 1 \le j \le r$
6. Compute $\phi\left(x^l\right) = \sigma\left(W^r h_{r-1} + b^r\right)$
7. Compute $o^l = W^{r+1}\phi\left(x^l\right) + b^{r+1}$
8. Compute $\delta_r^l = -2Cy^l\left(\max\left(1 - y^l o^l, 0\right)\right)\left(I\left\{1 > y^l o^l\right\}\right)$
9. Compute $\delta_j^l = \left[\left(W^{j+1}\right)^T \delta_{j+1}^l\right] \circ \sigma'\left(u_j^l\right), 1 \le j \le r$
10. Compute $\begin{cases} \frac{\partial L_N}{\partial W^j} = W^j - \sum_{l=1}^{N} \delta_{j-1}^l \left(h_{j-1}^l\right)^T, j = r+1 \\ \frac{\partial L_N}{\partial W^j} = \sum_{l=1}^{N} \delta_j^l \left(h_{j-1}^l\right)^T, 1 \le j \le r \end{cases}$
11. Compute $\frac{\partial L_N}{\partial b^j} = \sum_{l=1}^{N} \delta_j^l, 1 \le j \le r+1$
12. Update weights and biases: $W^j \leftarrow W^j - \eta\frac{\partial L_N}{\partial W^j}, b^j \leftarrow b^j - \eta\frac{\partial L_N}{\partial b^j}$
13. **End For**
14. **End For**

## 3. Deep Recurrent Kernel Mapping SVM (DRKMSVM) for Short Text Classification

RNN is good at processing sequence data for classification decision or regression estimation [36]. It is worth mentioning that there exists a strong correlation in text data, which belongs to the category of sequential data. In contrast to the conventional multi-layer perceptron neural networks, RNN is included in its own connections. It can process the current input vector based on the understanding of the context information, which is requisite to comprehend language. As a result, in this paper, kernel mapping is represented by RNN whose overall structure is shown in Figure 3.

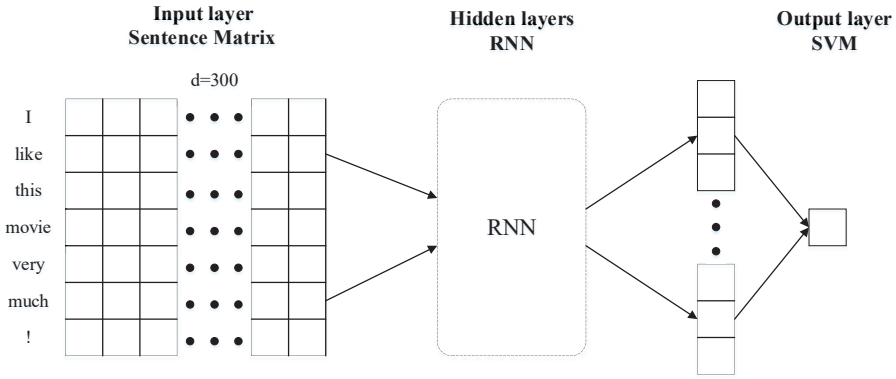


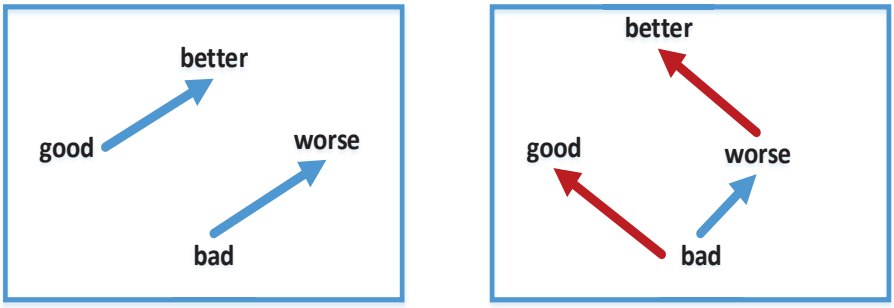

**Figure 3.** Schematic diagram of DRKMSVM model.

As can be seen from Figure 3, this model consists of three components. To begin with, word vector is used to represent the short texts. In addition, RNN is used in representation of kernel mapping, and finally the SVM is used for classification. To be specific, when the sentence length of the input is 7 and the word vector dimension is 300, the short texts will first of all be represented by DRKMSVM as a sentence matrix. Subsequently, with the help of RNN which extracts time-sequence features of the text, the eventual hidden layer neuron states are spliced into feature vector $\Phi(\boldsymbol{x}^l)$ which functions as the final representation of short text. In the end, the final representation will be input to the output layer, and classification is carried out by means of soft margin support vector machine. The output will be labeled as $+1$ or $-1$. In the following sections, the three components will be described in turn.

### *3.1. Representing the Short Text with Word Vector*

Word2vec is a statistical approach which can effectively learn word vectors from corpus [37]. By learning from a billion level corpus, this method can acquire high-quality word vectors at a lower time complexity. One of the features of Word2vec is that the syntactic and semantic similarity learned by it can be measured by distance, that is to say, words that are similar in the light of semantics or grammar can be detected by calculating their similarity. Figure 4 displays the schematic diagram of word vector similarity. Figure 4a stands for the offset of two pairs of words in order to clarify their corresponding comparative-level relationship, while Figure 4b refers to different projections. As far as the semantic similarity of word vectors is concerned, provided that cosine similarity is applied to measurement, it is possible to fine tune the similar words.

We employ a publicly available Word2vec to fine tune word vectors (The training data came from Google news, and the word vector dimension was 300). The results after fine-tuning is shown in Table 1. From Table 1, we can find that, before fine-tuning, the word "bad" is one of the four words with the highest similarity with "good", probably because they are both adjectives that are antonyms in terms of semantic meaning and almost equivalent grammatically. However, after fine-tuning, the word with higher similarity with "good" turned to be "nice", which indicates that in the corpus adopted in this paper, "good" is closer to "nice" since both of them convey similar emotions. Besides, changes of words with higher similarity with the word "not" indicate that in pre-training corpus, "not" is often collocated together with "do" and "did" to indicate negation. The reason accounting for the changes after fine-tuning may be ascribed to adversative relation. All in all, the fine-tuned word vector is more accurate and specific in task of short text classification.




(a) (b)

**Figure 4.** Word vector similarity diagram. (**a**) The offset of two pairs of words. (**b**) The offset of two pairs of words under different projections.

**Table 1.** The most similar word change before and after fine tuning.

| Word | Before Fine-Tuning | After Fine-Tuning |
|---|---|---|
| good | great<br>nice<br>terrific<br>decent | great<br>bad<br>terrific<br>decent |
| bad | lousy<br>horrible<br>crummy<br>lousy | good<br>terrible<br>horrible<br>lousy |
| funny | hilarious<br>witty<br>comical<br>sarcastic | hilarious<br>humorous<br>hilariously_funny<br>amusing |
| boring | dull<br>uninteresting<br>monotonous<br>pointless | dull<br>uninteresting<br>monotonous<br>bored |
| but | so<br>yet<br>although<br>too | although<br>though<br>because<br>so |
| not | do<br>neither<br>however<br>either | do<br>did<br>anymore<br>necessarily |

### 3.2. *Representing Kernel Mapping with BRNN*

Various RNN structures have been developed, such as Long-short Time Memory (LSTM) [38], Gated Recurrent Unit (GRU) [39], and Bidirectional RNN [33]. On account of the context information in the text, we use BRNN for calculation, and for nodes in each hidden layer, GRU is used for calculation. The structure of BRNN and GRU are demonstrated in Figures 5 and 6, respectively.

In Figure 5, $\boldsymbol{x}_i (0 \leq i \leq len)$ refers to the word vector of the $i$-th word in a sentence, and $\boldsymbol{y}_i (0 \leq i \leq len)$ represents the output value in accordance with $\boldsymbol{x}_i$. $A_i (0 \leq i \leq len)$ as well as $A_i^{'} (0 \leq i \leq len)$

is a single neuron in BRNN, where $A_i\ (0 \le i \le len)$ is responsible for delivering the forward state, and $A_i^{'}(0 \le i \le len)$ is in charge of the backward state. Notably, the final output value is co-determined by the forward state and the backward state.

In the forward process, forward state $\boldsymbol{s}_i(0 \le i \le len+1)$ is co-determined by $\boldsymbol{s}_{i-1}(0 \le i \le len+1)$ and the input $\boldsymbol{x}_i(0 \le i \le len)$, which can be calculated as $\boldsymbol{s}_i = f(\boldsymbol{U}\boldsymbol{x}_i + \boldsymbol{W}\boldsymbol{s}_{i-1})$; backward state is co-determined by $\boldsymbol{s}_i(0 \le i \le len)$ and the input $\boldsymbol{x}_i(0 \le i \le len)$, which can be calculated as $\boldsymbol{s}_i^{'} = f(\boldsymbol{U}^{'}\boldsymbol{x}_i + \boldsymbol{W}^{'}\boldsymbol{s}_{i+1}^{'})(0 \le i \le len)$; the eventual output is co-determined by $\boldsymbol{s}_i$ and $\boldsymbol{s}_i^{'}$, which can be calculated as $\boldsymbol{y}_i = g(\boldsymbol{V}\boldsymbol{s}_i + \boldsymbol{V}^{'}\boldsymbol{s}_i^{s})(0 \le i \le len)$. In other words, $A_0, A_1, ..., A_i, ..., A_{len}(0 \le i \le len)$ in Figure 5 stand for neuron at different times, while their internal structure and parameters are identical.

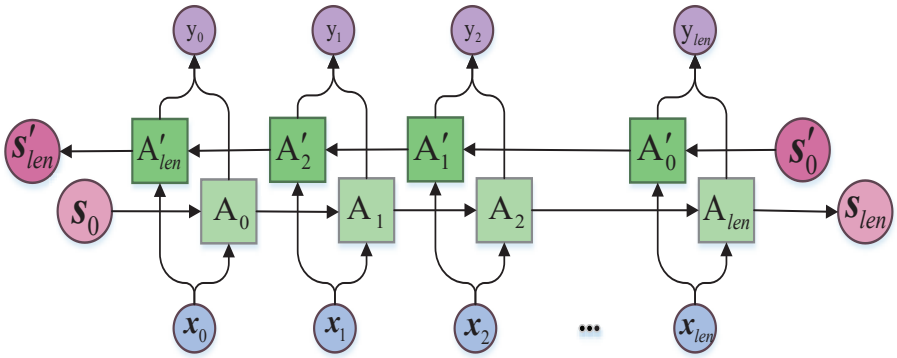



**Figure 5.** The structure of bidirectional recurrent neural network (BRNN).

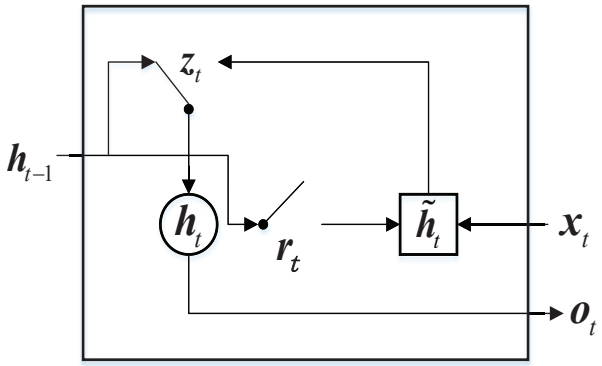



**Figure 6.** GRU cell.

In Figure 6, $\boldsymbol{z}_t$ and $\boldsymbol{r}_t$ refer to the update gate and the reset gate, respectively. Suppose that the inputs of the update gate and the reset gate are neuron state $\boldsymbol{h}_{t-1}$ at $t-1$ moment and word vector $\boldsymbol{x}_t$ at $t$ moment, they can be calculated as Equations (4) and (5) respectively.

$$\boldsymbol{r}_t = \sigma(\boldsymbol{W}_r \cdot [\boldsymbol{h}_{t-1}, \boldsymbol{x}_t]) \tag{4}$$

$$\boldsymbol{z}_t = \sigma(\boldsymbol{W}_z \cdot [\boldsymbol{h}_{t-1}, \boldsymbol{x}_t]) \tag{5}$$

where "[]" stands for concatenation operation of vector, "$\cdot$" is to multiply element-by-element, and $\sigma(\cdot)$ represents the sigmoid activation function. Likewise, the inputs of the hidden layer are neuron state $\boldsymbol{h}_{t-1}$ at $t-1$ moment and word vector $\boldsymbol{x}_t$ at $t$ moment. Thus, it can be calculated as follows:

$$\begin{cases} \tilde{\boldsymbol{h}}_t = g\left(\boldsymbol{W}_{\tilde{\boldsymbol{h}}} \cdot [\boldsymbol{r}_t * \boldsymbol{h}_{t-1}, \boldsymbol{x}_t]\right) \\ \boldsymbol{h}_t = (1 - \boldsymbol{z}_t) * \boldsymbol{h}_{t-1} + \boldsymbol{z}_t * \tilde{\boldsymbol{h}}_t \end{cases} \tag{6}$$

where $g(\cdot)$ represents the $tanh(\cdot)$ activation function. In the end, the neuron output can be calculated as:

$$f_t = \sigma(W_t^o \cdot h_t) \tag{7}$$

### 3.3. Classifying with SVM

Given that after using the BRNN mapping, the final short text obtained by the $l$-th sample $x^l$ is expressed as $y^{l,R}$, which is input to the soft-margin SVM to classify. At this point, the final output of DRKMSVM is expressed as:

$$o^l = W^o y^{l,R} + b^o \tag{8}$$

where $W^o$ and $b^o$ are the weight and bias of SVM, respectively. $y^{l,R}$ is generated by splicing the output results of neurons at all moments in the end, which is expressed as:

$$\phi(x^l|\theta) = y^{l,R} = \left[y_0^{l,R}\ y_1^{l,R}\ \cdots\ y_i^{l,R}\ \cdots\ y_{len}^{l,R}\right] \tag{9}$$

In the course of training, the target function of DRKMSVM is expressed as:

$$\begin{aligned} L_N = & \min_{\substack{W^j, b^j, \xi_l \\ 1 \le j \le r+1, 1 \le l \le N}} \frac{1}{2}||W^o||^2 + C\sum_{l=1}^{N}\xi_l^2 \\ s.t. & y^l(W^o y^{l,R} + b^o) \ge 1 - \xi_l, \xi_l \ge 0, l = 1, ..., N \end{aligned} \tag{10}$$

The problem of function constrained extremum demonstrated in Equation (10) is equivalent to the following problem of unconstrained optimization:

$$L_D = \min_{\theta} \frac{1}{2}||W^o||^2 + C\sum_{l=1}^{N}\left[max(1 - y^l \times (W^{r+1}\phi(x^l|\theta) + b^o), 0)\right]^2 \tag{11}$$

where $\theta = \{U, U', W, W', V, V', W_r, W_z, W_{\tilde{h}}, W_t^o\} \cup \{W^o, b^o\}$ represents all the parameters remain to be solved and $y^l$ is the label of $x^l$, whose value is $+1$ or $-1$. Later on, DRKMSVM will be trained with the back-propagation with time algorithm so as to obtain its weights and biases.

## 4. Experimental Results

In order to verify the effectiveness of DRKMSVM in short text classification, we conducted experiments on 5 public short text datasets and made comparisons with other well-known methods, including DNMSVM, CNN, SVM with radial basis function kernel (RBF-SVM) and NB. To further illustrate the performance of DRKMSVM, we analyzed the influence of the structure of recurrent network on the performance of DRKMSVM. All the experiments are completed under the framework of the TensorFlow [40], with a server of 64-bit windows 7 operating system, Intel® Xeon® E5-2643 processor and NVIDIA Tesla K40c GPU.

### 4.1. Datasets

Our experiments were conducted on the following 5 public datasets [41]:

- MR (Movie Review) (http://www.cs.cornell.edu/people/pabo/movie-review-data/): This dataset is often refered to as polarity dataset, it intends to divide movie reviews into positive or negative reviews.
- CR (Custom Review) (https://www.cs.uic.edu/~liub/FBS/sentiment-analysis.html#datasets/): This dataset intends to predict whether a review is good or bad.
- Subj (Subject) (http://www.cs.cornell.edu/people/pabo/movie-review-data/): It is known as subjectivity dataset and its categorization purpose lies in predicting whether the sentence is subjective or objective.

- MPQA (Multiple Perspective QA) (http://mpqa.cs.pitt.edu/): This dataset contains new articles from a wide variety. In this paper, we choose opinion polarity detection subtask of the MPQA dataset, it aims to determine whether an opinion sentence is positive or negative.
- TREC (Text Retrieval Conference) (https://cogcomp.seas.upenn.edu/Data/QA/QC/): This dataset is composed of 6 types of problems, which aims to predict which category of problem it belongs to.

The detailed information about these five datasets are demonstrated in Table 2, including their name, abbreviation, total sample number (TAN), class number (CN), average sentence length (ASL), and dictionary length (DL), and number of validation set sample (NVSS).

**Table 2.** Information of datasets.

| Name | Abbreviation | TSN | CN | ASL | DL | NVSS |
|---|---|---|---|---|---|---|
| Movie Review | MR | 9596 | 2 | 111 | 18,765 | 1000 |
| Custom Review | CR | 3397 | 2 | 93 | 5046 | 378 |
| Subject | Subj | 9000 | 2 | 125 | 17,913 | 1000 |
| Opinion polarity dataset | MPQA | 9546 | 2 | 18 | 5046 | 1060 |
| TREC QA | TREC | 5452 | 6 | 60 | 6083 | 500 |

Among the 5 datasets, only the TREC dataset has been divided into training set and testing set. For the other four datasets MR, CR, Subj, and MPQA, there is no standard partition of training set and testing set available. Therefore, 10-fold cross-validation is applied in the experiments for the above four datasets. That is we will randomly divide a data set 10 times, and each time the ratio of the training set and testing set is 9:1. The average result of 10 experiments will be taken as the final result.

### *4.2. Parameter Settings of DRKMSVM, DNMSVM, CNN and SVM*

Before applying DRKMSVM for short text classification, corresponding parameters need to be set. For DRKMSVM, the bidirectional structure is employed. The number of hidden nodes, learning rate, the size of mini-batch, dropout and penalty factors are shown in Table 3. Among them, the hyper-parameters in the experiments are selected by means of cross-validation on MR dataset, whose referential evaluation index is accuracy. The candidate set of hidden node number is $\{i \times 100 \,|i = 1, 2, 3, 4, 5\}$, the candidate set of learning rate is $\left\{10^i \,|i = -4, -3, -2, -1\right\}$, dropout is 0.5 in accordance with relevant papers, the candidate set of the size of mini-batch is 32, 64, 128, and the candidate set of penalty factor $C$ is $\left\{2^i \,|i = -3, -2, \cdots, 8\right\}$.

For DNMSVM, the number of hidden layers is set as 2 for testing. Therefore, the structure of DNMSVM can be expressed as $x - h_1 - h_2 - o$. Note that in our experiments, we rename DNMSVM with two hidden layers using the unified name DMKMSVM-2. Other parameters, such as the number of hidden nodes, learning rate, the size of mini-batch, dropout, and penalty factors are set in accordance with relevant papers. Among them, the hyper-parameters are selected by means of cross-validation on MR dataset, whose referential evaluation index is accuracy. The candidate set of hidden node number is $\{i{*}100|i = 1, ...., 5\}$, the candidate set of learning rate is $\left\{10^i \,|i = -4, \ldots, -1\right\}$, the candidate set of the size of mini-batch is {32,64,128}, the candidate set of penalty factor $C$ is $\left\{2^i \,|i = -3, -2, \ldots, 8\right\}$, and the dropout is 0.5. The parameters of DMKMSVM-2 used in the experiments are illustrated in Table 4.

**Table 3.** Hyper-parameter setting of DRKMSVM.

| Hyper-Parameter | Value |
|---|---|
| Number of hidden layers | 1 |
| Number of hidden notes | 300 |
| Learning rate | 0.001 |
| Mini-batch | 64 |
| Dropout | 0.5 |
| $C$ | 64 |

**Table 4.** Hyper-parameter setting of DMKMSVM-2.

| Hyper-Parameter | Value |
|---|---|
| Number of hidden notes | 100 |
| Learning rate | 0.001 |
| Mini-batch | 64 |
| Dropout | 0.5 |
| $C$ | 64 |

With regard to CNN, word vector is adopted, and its parameter settings are shown in Table 5. Notably, for RBF-SVM, the two parameters $C$ and $\gamma$ need to be determined by 10-fold cross-validation, where the candidate set of $C$ is $\{2^i | i = -3, -2, \ldots, 8\}$ and the candidate set of $\gamma$ is $\{2^i | -4, -3, \ldots, 5, 6, 7\}$. The final selection is shown in Table 6.

**Table 5.** Parameter settings of convolutional neural network (CNN).

| Hyper-Parameters | Value |
|---|---|
| Size of convolutional kernel | 3300,4300,5300 |
| Number of convolutional kernel | 128,128,128 |
| Learning rate | 0.001 |
| Minibatch | 64 |
| Dropout | 0.5 |

**Table 6.** Setting of parameters of RBF-SVM.

| Datasets | $\gamma$ | $C$ |
|---|---|---|
| MR | 0.1 | 8 |
| CR | 0.1 | 8 |
| MPQA | 0.1 | 8 |
| Subj | 0.01 | 64 |
| TREC | 0.01 | 64 |

### *4.3. Comparison with DMKMSVM-2, CNN, RBF-SVM and NB*

In this subsection, the performance of DRKMSVM on short text classification is compared with that of DMKMSVM-2, CNN, RBF-SVM and NB, where DRKMSVM and CNN adopt word vectors as input, while RBF-SVM and NB use TF-IDF as input. The bidirectional structure is employed in DRKMSVM. To make a quantitative evaluation, we adopt 4 commonly employed valuation indexes in text classification: Accuracy [42], Precision [43], Recall [44] and F1-score [45]. The experimental results are shown in Table 7, with bold numbers representing the best results.

From Table 7 we can see that although for the dataset CR and Subj, DRKMSVM is not always the best. The overall performance of DRKMSVM is superior to the other four methods with the highest average value of four metrics on the five datasets. The detailed comparisons in terms of the four indexes are as follows:

(1) In terms of the accuracy and the classification precision, we can see that for the dataset CR, the NB method achieves the best results, and for the dataset Subj, DMKMSVM-2 exceeds DRKMSVM slightly. However, for the other datasets MR, MPQA and TREC, DRKMSVM obtain the best results comparing with other four methods.

(2) In the light of the recall rate, for the dataset Subj, DRKMSVM is slightly lower than that of DMKMSVM-2. However, DRKMSVM achieves the best results on the other four datasets (MR, CR, MPQA and TREC), and it is much higher than that of DMKMSVM-2.

(3) In the light of $F_1-$score, for the dataset CR and Subj, DRKMSVM is lower than that of DMKMSVM-2. However, DRKMSVM achieves the best results on the other three datasets (MR, MPQA and TREC).

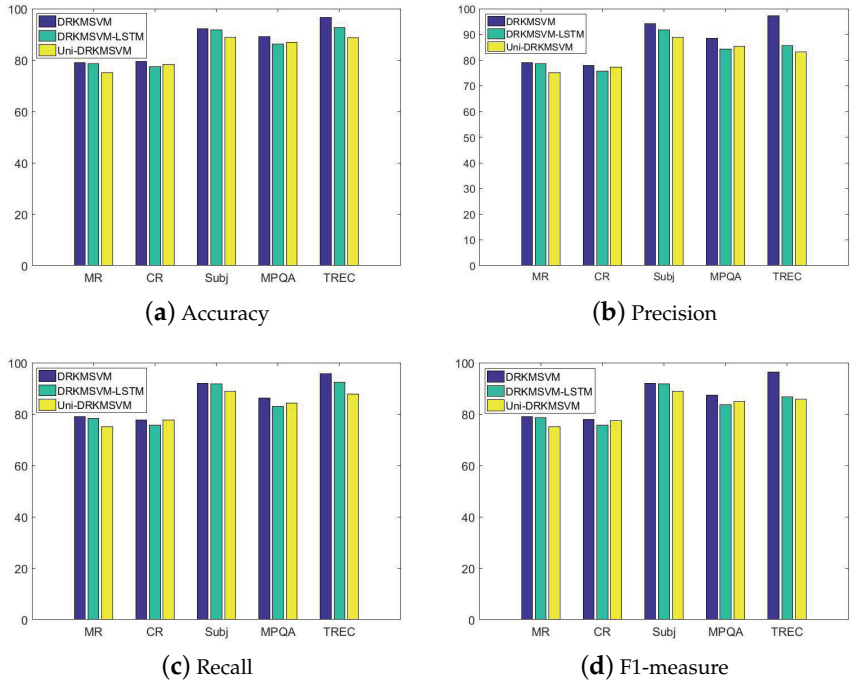
Based on the above analysis, it can be claimed that DRKMSVM has better classification performance than DMKMSVM-2, CNN, SVM and NB in most cases. The comparison results reveal that employing recurrent neural network to extract context feature mapping in text is conductive to enhancing the performance of short text classification.

**Table 7.** Accuracy, precision, recall and $F_1$-score of four algorithms on five datasets.

| Methods | Datasets | Accuracy (%) | Precision (%) | Recall (%) | $F_1$-Score (%) |
|---|---|---|---|---|---|
| DRKMSVM | MR | **78.91** | **78.91** | **78.96** | **78.94** |
| | CR | 79.37 | 77.96 | **77.79** | 77.89 |
| | Subj | 92.10 | 92.24 | 91.90 | 92.03 |
| | MPQA | **89.16** | **88.57** | **86.36** | **87.32** |
| | TREC | **96.61** | **97.26** | **95.64** | **96.36** |
| | Average | **87.23** | **86.99** | **86.13** | **86.51** |
| DMKMSVM-2 | MR | 78.83 | 78.90 | 78.77 | 78.78 |
| | CR | 80.77 | 79.93 | 77.79 | **78.49** |
| | Subj | **92.44** | **92.43** | **92.45** | **92.43** |
| | MPQA | 86.92 | 85.95 | 82.82 | 84.08 |
| | TREC | 87.20 | 89.40 | 83.776 | 85.88 |
| | Average | 85.23 | 85.32 | 83.12 | 83.93 |
| CNN | MR | 77.24 | 72.38 | 72.27 | 72.19 |
| | CR | 77.72 | 77.58 | 73.70 | 74.65 |
| | Subj | 90.83 | 90.78 | 90.83 | 90.79 |
| | MPQA | 87.66 | 86.14 | 85.68 | 85.93 |
| | TREC | 89.42 | 89.29 | 90.26 | 89.72 |
| | Average | 84.57 | 83.23 | 82.54 | 82.66 |
| SVM | MR | 77.21 | 77.22 | 77.20 | 77.28 |
| | CR | 79.47 | 78.11 | 76.77 | 77.27 |
| | Subj | 90.90 | 90.90 | 90.90 | 90.89 |
| | MPQA | 86.63 | 85.67 | 82.41 | 83.72 |
| | TREC | 81.60 | 83.95 | 81.89 | 82.64 |
| | Average | 83.16 | 83.17 | 81.83 | 82.36 |
| NB | MR | 73.66 | 73.65 | 73.66 | 73.65 |
| | CR | **83.60** | **81.38** | 76.57 | 78.36 |
| | Subj | 91.30 | 91.32 | 91.36 | 91.30 |
| | MPQA | 86.15 | 87.28 | 79.28 | 81.85 |
| | TREC | 77.80 | 67.07 | 66.12 | 66.10 |
| | Average | 82.50 | 80.14 | 77.39 | 78.25 |

### *4.4. Influence of the Structure of Recurrent Network on DRKMSVM*

In this subsection, we evaluate the influence of different recurrent structure on the performance of DRKMSVM. The experiments involve the BRNN with LSTM hidden layer neurons, denoted as DRKMSVM-LSTM, and the unidirectional RNN with GRU hidden layer neurons, denoted as Uni-DRKMSVM. All the experiments share the same parameters mentioned above in Table 3. The only difference is the recurrent structure, that is BRNN for DRKMSVM and RNN of Uni-DRKMSVM. Experimental results are shown in Figure 7 and Table 8.




(**a**) Accuracy (**b**) Precision

(**c**) Recall (**d**) F1-measure

**Figure 7.** Performance of DRKMSVM using different recurrent structures.

**Table 8.** Effect of the network structure for DRKMSVM.

| Methods | Datasets | Accuracy (%) | Precision (%) | Recall (%) | $F_1$-Score (%) |
|---|---|---|---|---|---|
| DRKMSVM | MR | 78.91 | 78.91 | 78.96 | 78.94 |
| | CR | 79.37 | 77.96 | 77.79 | 77.89 |
| | Subj | 92.10 | 92.24 | 91.90 | 92.03 |
| | MPQA | 89.16 | 88.57 | 86.36 | 87.32 |
| | TREC | 96.61 | 97.26 | 95.64 | 96.36 |
| | Average | **87.23** | **86.99** | **86.13** | **86.51** |
| DRKMSVM-LSTM | MR | 78.63 | 78.58 | 78.26 | 78.57 |
| | CR | 77.51 | 75.82 | 75.75 | 75.78 |
| | Subj | 91.70 | 91.69 | 91.67 | 91.68 |
| | MPQA | 86.33 | 84.39 | 82.97 | 83.62 |
| | TREC | 92.61 | 85.53 | 92.46 | 86.72 |
| | Average | 85.36 | 83.20 | 84.22 | 83.27 |
| Uni-DRKMSVM | MR | 75.07 | 75.06 | 75.06 | 75.06 |
| | CR | 78.30 | 77.20 | 77.79 | 77.44 |
| | Subj | 88.90 | 88.92 | 88.81 | 88.85 |
| | MPQA | 86.99 | 85.42 | 84.39 | 84.87 |
| | TREC | 88.78 | 83.21 | 87.88 | 85.86 |
| | Average | 85.36 | 83.20 | 84.22 | 83.27 |

From Figure 7 and Table 8, we can see that in terms of four classification indexes, the performance of DRKMSVM exceeded those of DRKMSVM-LSTM and Uni-DRKMSVM on five datasets, which means that the proposed DRKMSVM which uses BRNN structure with GRU neurons is more competent than unidirectional recurrent network and BRNN with LSTM neurons in terms of improving the performance of text classification. This is because in the process of interpreting language, human beings tend to take context into consideration, and bidirectional recurrent network can effectively obtain both the previous and the future features and thus achieve better effects than unidirectional recurrent network.

## 5. Conclusions

To deal with the problem of short text classification, we introduced a uniform framework of deep kernel mapping support vector machine (D*U*KMSVM). Based on this framework, we proposed a DRKMSVM to improve the performance of short text classification. The advantages of the DRKMSVM is that by applying the bidirectional recurrent neural network with GRU neurons to express the kernel mapping of SVM explicitly, it does not require the kernel trick to solve the parameters. Besides, bidirectional neural network can effectively obtain the context information in the short text, which is helpful for classification. Experimental results on five publicly available English datasets indicate that the in terms of four measurements, classification accuracy, precision, recall rate and $F_1-$score, the DRKMSVM are superior to those of DMKMSVM, CNN, RBF-SVM and NB in most instances. In the future, we will continue to explore how to design more reasonable recurrent neural network structure and neurons to further enhance its performance on short text classification.

**Author Contributions:** Z.L., T.Z. and Y.L. Conceptualization, and Funding Acquisition; Y.L. proposed the idea; H.K. performed the experiments; Z.L. analyzed the results and wrote the paper; T.Z. helped draw the figures and revised the clarity of the work; Z.L., H.K., T.Z., and Y.L. approved the final manuscript. All authors have read and agreed to the published version of the manuscript.

**Funding:** This work was supported by the National Natural Science Foundation of China (61806013, 61876010, 61906005); Chaoyang Postdoctoral Foundation of Beijing (2019zz-43); Program of College of Computer Science, Beijing University of Technology (2019JSJKY006); and Open project of key laboratory of oil and gas resources research, Chinese academy of sciences (KLOR2018-9).

**Conflicts of Interest:** The authors declare no conflict of interest.

## References

1. Ren, F.; Deng, J. Background Knowledge Based Multi-Stream Neural Network for Text Classification. *Appl. Sci.* **2018**, *8*, 2472. [CrossRef]
2. Nigam, K.; McCallum, A.K.; Thrun, S.; Mitchell, T. Text classification from labeled and unlabeled documents using EM. *Mach. Learn.* **2000**, *39*, 103–134. [CrossRef]
3. Zhang, L.; Duan, Q. A Feature Selection Method for Multi-Label Text Based on Feature Importance. *Appl. Sci.* **2019**, *9*, 665. [CrossRef]
4. Joulin, A.; Grave, E.; Bojanowski, P.; Mikolov, T. Bag of Tricks for Efficient Text Classification. *arXiv* **2016**, arXiv:1607.01759.
5. Li, F.; Yin, Y.; Shi, J.; Mao, X.; Shi, R. Method of Feature Reduction in Short Text Classification Based on Feature Clustering. *Appl. Sci.* **2019**, *9*, 1578. [CrossRef]
6. Forman, G. An extensive empirical study of feature selection metrics for text classification. *J. Mach. Learn. Res.* **2003**, *3*, 1289–1305.
7. Aggarwal, C.C.; Zhai, C. A Survey of Text Classification Algorithms. In *Mining Text Data*; Springer US: Boston, MA, USA, 2012; pp. 163–222.
8. Kim, S.B.; Han, K.S.; Rim, H.C.; Myaeng, S.H. Some effective techniques for naive bayes text classification. *IEEE Trans. Knowl. Data Eng.* **2006**, *18*, 1457–1466.
9. Sun, A. Short text classification using very few words. In Proceedings of the 35th International ACM SIGIR Conference on Research and Development in Information Retrieval, Portland, OR, USA, 12–16 August 2012; pp. 1145–1146.

10. Lin, Z.; Yan, L. A support vector machine classifier based on a new kernel function model for hyperspectral data. *GIS. Remote Sens.* **2016**, *53*, 85–101. [CrossRef]
11. Amari, S.; Wu, S. Improving support vector machine classifiers by modifying kernel functions. *Neur. Net.* **1999**, *12*, 783–789. [CrossRef]
12. Cassel, M.; Lima, F. Evaluating one-hot encoding finite state machines for SEU reliability in SRAM-based FPGAs. In Proceedings of the 12th International On-Line Testing Symposium, Lake of Como, Italy, 10–12 July 2006; pp. 1–6.
13. Zhang, Y.; Jin, R.; Zhou, Z.H. Understanding bag-of-words model: A statistical framework. *Int. J. Mach. Learn. Cybern.* **2010**, *1*, 43–52. [CrossRef]
14. Hinton, G.E.; Salakhutdinov, R.R. Reducing the dimensionality of data with neural networks. *Science* **2006**, *313*, 504–507. [CrossRef]
15. Faruqui, M.; Tsvetkov, Y.; Yogatama, D.; Dyer, C.; Smith, N. Sparse overcomplete word vector representations. *arXiv* **2015**, arXiv:1506.02004.
16. Mikolov, T.; Chen, K.; Corrado, G.; Dean, J. Efficient estimation of word representations in vector space. *arXiv* **2013**, arXiv:1301.3781.
17. Deng, W.W.; Peng, H. Research on a naive bayesian based short message filtering system. In Proceedings of the International Conference on Machine Learning and Cybernetics, Dalian, China, 13–16 August 2006; pp. 1233–1237.
18. Schneider, K.M. Techniques for improving the performance of naive bayes for text classification. In Proceedings of the International Conference on Intelligent Text Processing and Computational Linguistics, Mexico City, Mexico, 13–19 February 2005; pp. 682–693.
19. Zhao, W.; Meng, L.; Zhao, H.; Wang, C. Improvement and Applications of the Naive Algorithm. *Meas. Control. Technol.* **2016**, *35*, 143–147.
20. Khamar, K. Short text classification using kNN based on distance function. *Int. J. Adv. Res. Comput. Commun. Eng.* **2013**, *2*, 1916–1919.
21. Guo, G.; Wang, H.; Bell, D.; Bi, Y.; Greer, K. Using kNN model for automatic text categorization. *Soft Comput.* **2006**, *10*, 423–430. [CrossRef]
22. Shi, K.; Li, L.; Liu, H.; He, J.; Zhang, N.; Song, W. An improved KNN text classification algorithm based on density. In Proceedings of the International Conference on Cloud Computing and Intelligence Systems, Beijing, China, 15–17 September 2011; pp. 113–117.
23. Yin, C.; Xiang, J.; Zhang, H.; Wang, J.; Yin, Z.; Kim, J. A new SVM method for short text classification based on semi-supervised learning. In Proceedings of the 4th International Conference on Advanced Information Technology and Sensor Application, Harbin, China, 21–23 August 2015; pp. 100–103.
24. Song, G.; Ye, Y.; Du, X.; Huang, X.; Bie, S. Short text classification: A survey. *J. Multim.* **2014**, *9*, 635. [CrossRef]
25. Sanchez, A.V.D. Advanced support vector machines and kernel methods. *Neurocomputing* **2003**, *55*, 5–20. [CrossRef]
26. Hassan, A.; Mahmood, A. Efficient Deep Learning Model for Text Classification Based on Recurrent and Convolutional Layers. In Proceedings of the 16th IEEE International Conference on Machine Learning and Applications (ICMLA), Cancun, Mexico, 18–21 December 2017; pp. 1108–1113.
27. Tang, D.; Wei, F.; Yang, N.; Zhou, M.; Liu, T.; Qin, B. Learning sentiment-specific word embedding for twitter sentiment classification. In Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, Baltimore, MD, USA, 22–27 June 2014; pp. 1555–1565.
28. Kim, Y. Convolutional neural networks for sentence classification. In Proceedings of the International Conference on empirical Methods in Natural Language Processing, Doha, Qatar, 25–29 October 2014; pp. 1532–1543.
29. Er, M.J.; Zhang, Y.; Wang, N.; Pratama, M. Attention pooling based convolutional neural network for sentence modelling. *Inf. Sci.* **2016**, *373*, 388–403. [CrossRef]
30. Shen, T.; Zhou, T.; Long, G.; Jiang, J.; Pan, S.; Zhang, C. DiSAN: Directional self-attention network for rnn/cnn-free language understanding. *arXiv* **2017**, arXiv:1709.04696.
31. Zhou, C.; Sun, C.; Liu, Z.; Lau, F. A C-LSTM Neural Network for Text Classification. *Compos. Sci.* **2015**, *1*, 39–44.
32. Olabiyi, O.; Martinson, E.; Chintalapudi, V.; Guo, R. Driver Action Prediction Using Deep (Bidirectional) Recurrent Neural Network. *arXiv* **2017**, arXiv:1706.02257.

33. Schuster, M.; Paliwal, K.K. Bidirectional recurrent neural networks. *IEEE Trans. Signal Process.* **1997**, *45*, 2673–2681. [CrossRef]
34. Li, Y.; Zhang, T. Deep neural mapping support vector machines. *Neural Netw.* **2017**, *93*, 185–194. [CrossRef]
35. Hearst, M.A.; Dumais, S.T.; Osuna, E.; Platt, J.; Scholkopf, B. Support vector machines. *IEEE Intell. Syst. Their Appl.* **1998**, *13*, 18–28. [CrossRef]
36. Chung, J.; Gulcehre, C.; Cho, K.H.; Bengio, Y. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv* **2014**, arXiv:1412.3555.
37. Goldberg, Y.; Levy, O. Word2vec Explained: Deriving Mikolov et al.'s negative-sampling word-embedding method. *arXiv* **2014**, arXiv:1402.3722.
38. Lipton, Z.C.; Berkowitz, J.; Elkan, C. A Critical Review of Recurrent Neural Networks for Sequence Learning. *arXiv* **2015**, arXiv:1506.00019.
39. Cho, K.; Merrienboer, B.; Gulcehre, C.; Bahdanau, D.; Bougares, F.; Schwenk, H.; Bengio, Y. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. *arXiv* **2014**, arXiv:1406.1078.
40. Abadi, M.; Agarwal, A.; Barham, P.; Brevdo, E.; Chen, Z.; Citro, C.; Corrado, G.; Davis, A.; Dean, J.; Devin, M.; et al. TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems. *arXiv* **2016**, arXiv:1603.04467.
41. Kim, Y. Convolutional neural networks for sentence classification. *arXiv* **2014**, arXiv:1408.5882.
42. Story, M.; Congalton, R.G. Accuracy assessment: A user's perspective. *Photogramm. Eng. Remote Sens.* **1986**, *52*, 397–399.
43. Powers, D.M. Evaluation: From precision, recall and F-measure to ROC, informedness, markedness and correlation. *J. Mach. Learn. Technol.* **2011**, *2*, 37–63.
44. Davis, J.; Goadrich, M. The relationship between Precision-Recall and ROC curves. In Proceedings of the 23rd International Conference on Machine Learning, Pittsburgh, PA, USA, 25–29 June 2006; pp. 233–240.
45. Sammut, C.; Webb, G.I. F1-Measure. In *Encyclopedia of Machine Learning and Data Mining*; Springer US: Boston, MA, USA, 2017; p. 497.