Word Combination Kernel for Text Categorization

Lujiang Zhang[1], Xiaohui Hu[2], Shiyin Qin[1]
[1]School of Automation Science and Electrical Engineering
Beijing University of Aeronautics & Astronautics
Beijing 100191, China
[2]Institute of Software
Chinese Academy of Sciences
Beijing 100190, China
zhanglujiang@163.com, qsy@buaa.edu.cn, hxh@iscas.ac.cn

Journal of Digital
Information Management

ABSTRACT: *We proposed a novel kernel for text categorization. This kernel is an inner product in the feature space generated by all word combinations of specified length. A word combination is a collection of different words co-occurring in the same sentence. The word combination of length k is weighted by the k-th root of the product of the inverse document frequencies (IDF) of its words. A computationally simple and efficient algorithm was proposed to calculate this kernel. By restricting the words of a word combination to the same sentence and considering multi-word combinations, the word combination features can capture similarity at a more specific level than single words. By discarding word order, the word combination features are more compatible with the flexibility of natural language and the dimensionality this kernel can be reduced significantly compared to the word-sequence kernel. We conducted a series of experiments on the Reuters-21578 dataset and 20 Newsgroups dataset. This kernel consistently achieves better performance than the classical word kernel and word-sequence kernel on the two datasets. We also assessed the impact of word combination length on performance and compared the computing efficiency of this kernel to those of the word kernel and word-sequence kernel.*

Categories and Subject Descriptors
I.2.7 Natural Language Processing Text analysis: **I.7.1 [Document and Text Editing]**: Document management

General Terms: Text categorization, Kernal processing, word combination, Natural Language Processing

Keywords: Machine Learning, Support Vector Machine, Kernel Methods, Text Classification, Word-Combination Kernel

Received: 23 December 2011, Revised 1 March 2012, Accepted 5 March 2012

1. Introduction

The Support Vector Machine (SVM) is a state of the art machine learning technique which has achieved great success in many domains. It is also very promising for text categorization [11] [7] [30] and has become a popular choice in this area. The effect of Support Vector Machines depends largely on the choice of kernel. The first and most commonly used kernel for text classification is the *word kernel* [11], which represents documents using the *bag-of-words* model [19], then in conjunction with general purpose kernels (linear, polynomial, RBF). The *bag-of-words* model assumes that the words in a document are independent each other, and their relative positions have no effect on text classification. Only the word frequencies (possibly with additional weighting and normalization) are used to represent documents, while the information regarding word positions is totally discarded.

In [5] the authors proposed the *string kernel*, the first significant departure from the *bag-of-words* model. In *string kernel*, features are not word frequencies, but all possible ordered subsequences of characters occurring in documents. The similarity is assessed by the number of matching subsequences shared by two documents. [2] proposed the *word-sequence kernel* which extends the *string kernel* to process documents as word sequences. This approach greatly reduces the average length of symbols per document, which yields a significant improvement in computing efficiency. Moreover, matching word sequences allows working with more linguistically meaningful symbols.

There are still some issues with the *word-sequence kernel. Word-sequence kernel* usually confronts with very high dimensional and sparse feature space which hinders the effective training of kernel machines. In addition, natural language is flexible and considering word order is not helpful to text classification. [2] have proved by experiments that taking word order into account has very little effect on performance. [15] found that phrases (include n-grams, sequences of words, noun phrases such as named entities and other complex nominals) are not adequate to improve text classification accuracy, while elementary textual representations based on words is very effective.

We proposed a novel kernel called *word-combination kernel*. In this kernel we use the word combinations as features rather than single words or word sequences. The word combination is a collection of different words without order relations co-occurring in the same sentence. The feature space of this kernel is generated by all word combinations of specified length and this kernel is an inner product defined in this feature space. Documents usually have sparse representations in feature space. Reducing the feature dimensions of documents is helpful to alleviate the sparseness of feature representations. By discarding word order, the dimensionality of this kernel is significantly reduced compared to the *word-sequence kernel*. On the other hand, restricting the words of a word combination to the same sentence can somewhat retain a bit information regarding word positions, and the multi-word combination features can capture similarity at a more specific level than single words.

The remainder of this paper is organized as follows. In Section 2 we briefly introduce the related work. In Section 3 we give a detailed description of the *word-combination kernel*. Section 4 presents the experimental results and evaluation of the *word-combination kernel*. Finally, we conclude this paper in Section 5.

2. Related Work

A number of machine learning techniques have been applied to text categorization. A comprehensive survey about machine learning techniques for text categorization can be found in [22]. We focused on kernel methods for text classification in this paper. The effect of Support Vector Machines mainly depends on the choice of kernel. Support Vector Machines are very universal learners which can be combined with various kernels. The general kernels such as linear, polynomial and Gaussian RBF kernels [11] [13] have been used for text classification. [5] proposed the *Latent Semantic Kernels* based on the latent semantic indexing. Though having obtained good performance, these conventional kernels are based on the *bag-of-words* model and inherit its intrinsic drawbacks. Some researchers inject lexical dependencies [17] or semantic relations [29] into the vector representations of documents as an extension to the standard *bag-of-words* approach.

[14] proposed the *string kernel*, the first significant departure from the *bag-of-words* model. [2] proposed the *word-sequence kernel* to extend the *string kernel* to process documents as word sequences. Then the *factored sequence kernel* [3] was proposed as an extension of *sequence kernels* to the case where the symbols that define the sequences have multiple representations. *Word-sequence kernel* proves more effective than *string kernel*, especially in computing efficiency and using the standard linguistic preprocessing techniques. To resolve the issue of poor computational efficiency, the suffix-tree-based string kernel [26] and suffix-array-based string kernel [25] are proposed to quicken the calculation of *string kernel*. [24] proposed a statistical feature selection method which can be embedded into the *word-sequence kernel* to select significant features automatically.

Some researchers proposed syntactic and semantic kernels for text classification [23] [16] [28]. But only when the text categorization tasks are linguistically complex, such as classification in Question Answering (QA), syntax and semantics may play a relevant role [10] [16] [27]. Apparently promising syntactic and semantic structures have been shown inadequate for conventional text categorization tasks [1] [8] [12] [15].

3. Word-Combination Kernel

In this section we describe the details of *word-combination kernel*. The key point of this kernel is the use of word combination features. The word combination is a collection of different words co-occurring in the same sentence. The feature space of this kernel is generated by all word combinations of specified length. Note that there are no order relations between the words of a word combination. The basic idea of *word-combination kernel* is to measure the similarity between two documents by the number of word combinations they share in common. The more common word combinations they share, the more similar they are. The word combination of length k is weighted by the k-th root of the product of the inverse document frequencies (IDF) [20] of its words. The feature component corresponding to a word combination is the sum of weights over all occurrences of this word combination.

In general, natural language is flexible. One can often use several ways to express a meaning. And several words can be combined together to express a topic, but their orders may be changed. For example, "*give Mary a pie*" and "*give a pie to Mary*" express the same meaning; the phrases "seller of book" and "book seller" are equivalent to each other. We think that word combinations are more effective to measure the similarity between documents than word sequences. Documents usually have sparse representations in feature space. By discarding word order we can significantly reduce the feature dimensions of documents to obtain a more compact feature space. Besides, by restricting the words of a word combination to the same sentence and considering multi-word combinations, the word combination features can capture similarity at a more specific level than single words.

Conceptually, this kernel is a special case of convolution kernels [9], so we have the following definition of convolution kernel style.

3.1 Definition

Let Σ be a finite vocabulary. A document d is composed of n successive sentences, $d = s_1 s_2 ... s_n$. A sentence $s_i = \{s_{i_1}, s_{i_2}, ..., s_{i_{|s_i|}}\}(s_{i_j} \in \Sigma)$ is regarded as a collection of words with out considering word order. A word combination $u=\{u_1, u_2,...u_{|u|}\}(u_j \in \Sigma\ )$ is a collection of different words co-occurring in a sentence. We say u is a word combination of document d if and only if there exists at least a sentence s_i in document d satisfying $u \subseteq s_i$, and we use the short-hand notation $u \subseteq d\ [s_i]$ to denote it. We denote by Ω^n the set of all word combinations of length n.

We now define the feature space $F_n = R^{\Omega^n}$. The mapping ϕ for a document $d = s_1 s_2 ... s_n$ is given by defining the coordinate $\phi_u(d)$ for each $u \in \Omega^n$, which is $\phi : d \mapsto (\phi_u(d))_{u \in \Omega^n}$ We define:

$$\phi_u(d) = \sum_{s_i : u \subseteq d[s_i]} \lambda_u^d = \sum_{s_i : u \subseteq d[s_i]} (\lambda_{u_1}^d \lambda_{u_2}^d ... \lambda_{u_{|u|}}^d)^{\frac{1}{|u|}} \quad (1)$$
$$u \in \Omega^n$$

In Equation (1), $\lambda_u^d = (\lambda_{u_1}^d \lambda_{u_2}^d ... \lambda_{u_{|u|}}^d)^{\frac{1}{|u|}} (u_j \in u)$ is the weight of word combination u in document d, and $\lambda_{u_j}^d$ is the inverse document frequency (IDF) of the word u_j in document d. $\phi_u(d)$ is the sum of weights over all occurrences of the word combination u in document d.

In the feature space $F_n = R^{\Omega^n}$, the *word-combination kernel* is defined as the following Equation:

$$\begin{aligned} K_n(d_1, d_2) &= \sum_{u \in \Omega^n} \langle \phi_u(d_1), \phi(d_2) \rangle \\ &= \sum_{u \in \Omega} n \sum_{s_i : u \subseteq d_1[s_i]} \lambda_u^{d_1} \sum_{s_j : u \subseteq d_2[s_j]} \lambda_u^{d_2} \\ &= \sum_{u \in \Omega} n \sum_{s_i : u \subseteq d_1[s_i]} \ \sum_{s_j : u \subseteq d_2[s_j]} \\ &\quad (\lambda_{u_1}^{d_1} ... \lambda_{u_{|u|}}^{d_1})^{\frac{1}{|u|}} (\lambda_{u_1}^{d_2} ... \lambda_{u_{|u|}}^{d_2})^{\frac{1}{|u|}} \end{aligned} \quad (2)$$

$K_n(d_1,d_2)$ satisfies Mercer's conditions [21] because it is an inner product defined in the feature space $F_n = R^{\Omega^n}$. After the kernel has been computed we need to normalize it to remove any bias introduced by the document length. We use the L2-Norm to normalize the implicit feature vectors:

$$\begin{aligned} \hat{K}(d_1, d_2) &= \langle \hat{\phi}(d_1), \hat{\phi}(d_2) \rangle = \left\langle \frac{\phi(d_1)}{\|\phi(d_1)\|_2}, \frac{\phi(d_2)}{\|\phi(d_2)\|_2} \right\rangle \\ &= \frac{K(d_1,d_2)}{\sqrt{K(d_1,d_1) K(d_2,d_2)}} \end{aligned} \quad (3)$$

In Equation (1), we use the *k*-th root operator to make the weights of word combinations of different lengths have the same order of magnitude. This is helpful to combine the *word-combination kernels* with different feature length. For the word weighting, we don't consider the use of term-frequency, because the summation operation in Equation (1) has taken into account the word combination frequencies and the word frequencies can be embodied in the word combination frequencies. If we use the composite TF x IDF weight [20] instead of the inverse document frequency (IDF), the weighting system will contribute some redundant information about word frequency which can negatively bias the computed similarity.

Documents usually have very sparse feature representations in feature space. Reducing the dimensions of feature space is helpful to alleviate the sparseness of feature representations and obtain a more compact feature space. For a specified feature length n, the dimensionality of *word-combination kernel* is $C_n^{|\Sigma|}$ (C_y^x is the combination operator and Σ is a vocabulary), while that of the *word-sequence kernel* is $|\Sigma|^n$. The former is much lower than the latter because it satisfies

$$\frac{C_n^{|\Sigma|}}{|\Sigma|^n} = \frac{1}{n!} \frac{|\Sigma|...(|\Sigma|-n+1)}{|\Sigma|^n} < \frac{1}{n!}.$$

3.2 Combining kernels of different feature lengths

In general, word combinations of any length can make a contribution to the similarity between documents. So it is necessary to combine the kernels with different feature lengths. We proposed a linear combination formulation to combine the *word-combination kernels* with feature lengths from 1 to a fixed n, defined as follows:

$$K'_n(d_1, d_2) = \sum_{i=1}^n w_i K_i(d_1, d_2) \quad (3)$$

Kernels of different lengths should be normalized independently before being combined. If the running time is not concerned, we can obtain the optimized weighting parameters $w_i = (i = 1,...\ ,n)$ using the multiple kernels learning approach [18] or the *cross-validation* with *grid-search*. In actual practice we can simply set $w_i = i$. Though this may not be the optimal solution in theory, it can still obtain a good performance. Intuitively speaking, the importance of a word combination is proportional to its length. In Equation (1), the *k*-th root operator is used to make the weights of word combinations of different lengths have the same order of magnitude, which ensures the linear combination of *word-combination kernels* with different feature lengths makes sense.

3.3 The *sentence-intersections* between documents

A *sentence-intersection* is the intersection of a pair of sentences between two documents. Suppose that there are two sentences $s_1 = \{s_{1_1}, s_{1_2}, ..., s_{1_{|s_1|}}\}$ in document d_1 and $s_2 = \{s_{2_1}, s_{2_2}, ..., s_{2_{|s_2|}}\}$ in document d_2, then we have the *sentence-intersection* $si = s_1 \cap s_2$. The *sentence-intersections* are used to generate the common word combinations between two documents by using the combination generation algorithm. For example, from the *sentence-intersection* $si = \{w_1, w_1, ..., w_{|si|}\}$, we can generate the two-word common combinations: $\{w_1, w_2\}, \{w_1, w_3\}, ..., \{w_{|si|-1}, w_{|si|}\}$.

According to our statistic analysis, the distributions of *sentence-intersections* between documents in the Reuters-21578 dataset and 20 Newsgroups dataset are displayed in Tables 1 and 2. Table 1 displays the ratios of empty and non-empty *sentence-intersections*. Table 2 displays the ratios of *sentence-intersections* of different lengths among the non-empty *sentence-intersections*. The distributions of *sentence-intersections* between documents of the same categories and of different categories are listed separately. From the results in Tables 1 and 2, we can find three points. First, the distribution of *sentence-intersections* between documents of the same categories is distinctly different from the distribution of *sentence-intersections* between documents of different categories. Second, most of the *sentence-intersections* between documents are empty and the distribution of long *sentence-intersections* (whose length is greater than 2) is very sparse. Finally, the distribution of *sentence-intersections* in the Reuters-21578 dataset is distinctly different to the distribution of *sentence-intersections* in the 20 Newsgroups dataset.

	Empty	**Non-Empty**
Reuters-21578		
Ratio (of the same categories)	76.0%	24.0%
Ratio (of different categories)	88.6	11.4%
20 Newsgroups		
Ratio (of the same categories)	95.8%	4.2%
Ratio (of different categories)	98.2%	1.8%

Table 1. The ratios of empty and non-empty *sentence-intersections* between documents of the same categories and of different categories. The results are obtained after preprocessing

	n = 1	**n = 2**	**n = 3**	**n > 3**
Reuters-21578				
Ratio (of the same categories)	71.9%	18.3%	5.0%	4.8%
Ratio (of different categories)	85.9%	12.0%	1.7%	0.4%
20 Newsgroups				
Ratio (of the same categories)	89.6%	8.4%	1.2%	0.8%
Ratio (of different categories)	93.9%	5.8%	0.27%	0.03%

Table 2: The ratios of *sentence-intersections* of different lengths among the non-empty *sentence-intersections* between documents of the same categories and of different categories. The results are obtained after preprocessing

3.4 Algorithm

In this section we describe the algorithm of *word-combination kernel*. Before the calculation of this kernel, we need to preprocess the input documents. After preprocessing each document is converted into a list of sentences. We use the Equation (4) to compute the *word-combination kernel*.

To compute the *word-combination kernel*, the key step is to search all of the *sentence-intersections* between documents. Then the *sentence-intersections* are used to generate the common word combinations between documents by using the combination generation algorithm. According to our analysis, the main computing time of this kernel is consumed in searching the *word-intersections* between two documents. In order to reduce the searching time, we designed a hash table structure (Figure 1) for documents to quicken the searching operation. The key of this hash table is a word and the corresponding element is the list of sentences that contain this word. With the help of this structure, we proposed an algorithm which can find the sentences containing a specific word in $o(1)$ time. Before calculating the kernel values between documents, we first convert all documents into the form of hash tables as the Figure 1. The algorithm is as follows.

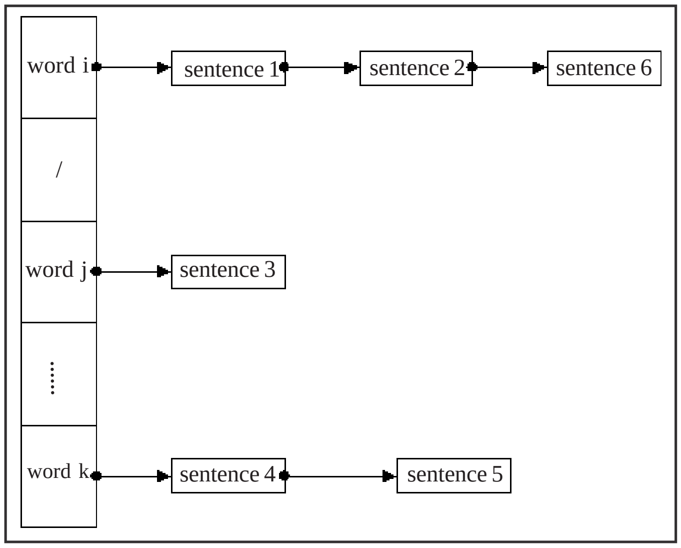

Figure 1. The hash table structure for documents

Algorithm 1: Converting each document into a hash table

```
Input:
$d$: The input document;
Output:
$d^H$: The output document in the form of hash table;
Procedure:
Hash table $d^H \leftarrow \varnothing$;
for each word $w_k$ in document $d$
  if $w_k$ exists in $d^H$
    List $L \leftarrow d^H[w_k]$;
    Add the sentence index of $w_k$ into $L$;
    $d^H[w_k] \leftarrow L$;
  else
    List $L \leftarrow \varnothing$;
    Add the sentence index of $w_k$ into $L$;
    $d^H[w_k] \leftarrow L$;
  end if
end for
```

Algorithm 2: Computing the kernel values with feature lengths from 1 to *n*

```
Input:
$d_1^H$, $d_2^H$: The two input documents in the form of hash tables;
$n$: The maximal word combination length;
Output:
$K[1…n]$: Kernel values with feature lengths from 1 to $n$;
Procedure:
$K[1…n] \leftarrow \{0,…,0\}$;
$s_1 \leftarrow$ The number of sentences of document $d_1^H$;
$s_2 \leftarrow$ The number of sentences of document $d_2^H$;
Sentence-intersection array $SI\,[1...s_1,1...s_2] \leftarrow \varnothing$;
for each key $w_k$ in $d_1^H$
  if $w_k$ exists in $d_2^H$
    List $L1 \leftarrow d_1^H[w_k]$;
    List $L2 \leftarrow d_2^H[w_k]$;
    for each sentence index $i$ in $L1$
      for each sentence index $j$ in $L2$
        Add $w_k$ to $SI\,[i, j]$ ;
      end for
    end for
  end if
end for
for each $s_i$ in $SI\,[1...s_1,1...s_2]$
  if $si \neq \varnothing$
    for $i \leftarrow 1$ to $n$
        $WC^i \leftarrow$ {The word combinations of length $i$
          generated from the sentence-intersection $si$};
      for each $WC_k^i$ in $WC^i$
        $K[i] \leftarrow K[i] + \lambda^{d_1}_{WC_k^i} * \lambda^{d_1}_{WC_k^i}$;
      end for
    end for
  end if
end for
```

In this algorithm, we use the combination generation algorithm to generate the common word combinations of specified length from the *sentence-intersection*. If we only generate the two-word combinations from the *sentence-intersection*, it can be easily realized by two nested loops. We denote by WC_k^i the *k*-th word combination of length i generated from a *sentence-intersection* si and denoted by $\lambda^{d_1}_{WC_k^i}$ the weight of WC in document d, then the partial inner product value between documents d_1 and d_2 corresponding to this *sentence-intersection* is $\sum_{k=1}^{C_i^{|si|}} \lambda^{d_1}_{WC_k^i} * \lambda^{d_2}_{WC_k^i}$ (C_y^x is the combination operator).

The computational complexity of this algorithm is $o(2|d_1| + 2|d_2| + s_1 s_2(C_1^{l_{max}} + 2C_2^{l_{max}} + nC_n^{l_{max}}))$. s_1 and s_2 are the number of sentences of documents d_1 and d_2 respectively. n is the maximal word combination length. l_{max} is the maximal length of *sentence-intersections* between documents d_1 and d_2. C_y^x is the combination operator. This computational complexity is upper bound and the actual complexity is lower than it because of the sparse distribution of long *sentence-intersections*. According to our statistic analysis (See Tables 1 and 2), most of the *sentence-intersections* between documents are empty and more than 95% of *sentence-intersections* between documents have a length less than or equal to 3. So the actual computational complexity is not higher than $o(2|d_1| + 2|d_2| + s_1 s_2(C_1^3 + 2C_2^3 + nC_n^3)) \leq o(2|d_1| + 2|d_2| + s_1 s_2(C_1^3 + 2C_2^3 + 3C_3^3)) = o(2|d_1| + 2|d_2| + 12\, s_1 s_2)$. Because s_1 and s_2 is much smaller than $|d_1|$ and $|d_2|$, this complexity is not high and roughly comparable to the linear complexity.

4. Experiments

In this section we describe our experiments. The objectives of these experiments are to

- observe the impact of word combination length on classification performance,
- compare the classification performance of this kernel to those of the *word kernel* and *word-sequence kernel*,
- compare the computing efficiency of this kernel to those of the *word kernel* and *word-sequence kernel*.

The Equation (4) is used to compute the *word-combination kernel* and we set the weighting parameters $w_i = i$ ($i = 1,...,n$). The experiments were conducted using the libSVM [4]. We use the SVM of *C*-SVC type. The 3-fold *cross-validation* is used to select the optimized value of C. We use the F_1 score as the performance evaluation and present the micro-averaged and macro-averaged scores as the synthetic measure of performance over all categories. For the *word kernel*, we use the linear kernel rather than the RBF kernel. As [30] have found, our experimental results also showed that the linear kernel can obtain a slightly better result than the RBF kernel in word space.

4.1 Dataset

We applied the Reuters-21578 dataset and 20 Newsgroups dataset to our experiments. The Reuters-21578 dataset was compiled by David Lewis in 1987 which is available at: http://www.research.att.com/lewis. We use the *"ModeApte"* split of the Reuters-21578 dataset. It comprises 9603 training and 3299 test documents which had been classified into 118 categories. The eight most frequent categories *"earn"*, *"acq"*, *"money"*, *"grain"*, *"crude"*, *"trade"*, *"interest"* and *"ship"* are selected as our target data set. We removed the overlapped test documents, but retained the overlapped training documents. Then the eight categories are summarized in Table 3. All training and test documents contained in the eight categories are used to train and test a SVM classifier.

Category	# training set	# test set
earn	2877	1083
acq	1650	709
money	538	130
grain	433	133
crude	389	142
trade	369	103
interest	347	87
ship	197	43

Table 3. Number of training/test examples after removing the overlapped test documents

The 20 Newsgroups data set is a collection of approximately 20,000 newsgroup documents. It was originally collected by Ken Lang which is available at: http://people.csail.mit.edu/jrennie/20Newsgroups/. The data is partitioned evenly across 20 different newsgroups, each corresponding to a different category. The headers of documents are removed. We repeatedly perform 10 runs for each algorithm, and evaluate the performance of a classifier by averaging the results over 10 runs. For each run we randomly selected 300 training documents and 100 test documents from each category as a target data set.

4.2 Data preprocessing

The preprocessing includes sentence boundary determination, stop word removal, stemming and computing the weight of words. After preprocessing, the Reuters-21578 dataset contains 26384 unique words, with average 66 words and 6.1 sentences per document; while the 20 Newsgroups data set contains average 72342 unique words, with average 118 words and 16.3 sentences per document. All documents are converted into the form of sentence list.

In order to extract the common word combinations, we need to determine the sentence boundaries in documents. Stop word removal filters out the words which are generally regarded as 'functional words' and do not carry meaning. We removed the punctuations and words occurring in a stop list. Stemming (Hull, 1996) is used to convert the words with a common root into a single stem. To simplify and quicken the stemming process, only the important singular/plural regularization was conducted to transform the plural noun into singular noun.

For the *word-combination kernel*, we use the inverse document frequency $idf_k = log\ \frac{N}{df_k}$ [20] to weight a word w_k. Here df_k is the number of documents containing the word w_k ,and N is the total number of documents in the data set. We use the L2-Norm to normalize the inverse document frequencies of each document: $idf_k \rightarrow \frac{idf_k}{\sqrt{\Sigma_i (idf_i)^2}}$

For the *word kernel* [11] and *word-sequence kernel* [2], we will strictly follow the weighting system proposed by their researchers.

4.3 Experimental results

In this section, we describe the experimental results. All experiments were conducted on the Reuters-21578 dataset and 20 Newsgroups dataset. In subsection 4.3.1 we observe the impact of word combination length on classification performance. In subsection 4.3.2 we compare the classification performance of this kernel to those of the *word kernel* and *word-sequence kernel*. In subsection 4.3.3 we compare the computing efficiency of this kernel to those of the *word kernel* and *word-sequence kernel*. The experimental results for the Reuters-21578 dataset are obtained with one run of the algorithm using the results for the 20 Newsgroups dataset are averaged over 10 runs of the algorithm using randomly selected 300 training documents and 100 test documents from each category.

4.3.1 Impact of word combination length on performance

We varied the feature length of *word-combination kernels* from 2 to 4 to assess the impact of word combination length on performance. The micro-averaged and macro-averaged F_1 scores varying with the word combination length are provided in Table 4.

From the Table 4, one can see that the micro-averaged and macro-averaged F_1 scores of *word-combination kernel* consistently decrease with the increase of word combination length on the Reuters-21578 dataset and 20 Newsgroups dataset. The *word-combination kernel* with length $n = 2$ obtains the best performance. This result can be intuitively explained by the sparse distribution of long common word combinations (whose length is greater than 2) between documents which can be deduced from the Table 2. When long word combinations are taken into account the precision increases, but the loss in recall more than offsets the increase, so the F_1 scores decrease.

4.3.2 The comparison of classification performance

We compare the classification performance of this kernel to those of the *word kernel* and *word-sequence kernel* on the Reuters-21578 dataset and 20 Newsgroups dataset. Both the performance for each category and the synthetic measure of performance over all categories are presented. Table 5 and Table 6 display the experimental results for the Reuters-21578 dataset and 20 Newsgroups dataset respectively. We use the boldface to mark the best result for each category. From the results one can see that the *word-combination kernel* consistently achieves good performance on the two datasets. On the Reuters-21578 dataset *word-combination kernel* obtains seven best results for the eight categories. On the 20 Newsgroups dataset *word-combination kernel* obtains fifteen best results for the twenty categories. The micro-averaged and macro-averaged F_1 scores of the *word-combination kernel* are also better than those of the *word kernel* and *word-sequence kernel*.

To further compare the performance of the three kernels, we applied a statistical significance test to the experimental results. The macro sign test (S-test) proposed by [30] is used. Table 7 shows the test results. The results in Table 7 provide clear evidence that the *word-combination kernel* outperforms the *word kernel* and *word-sequence kernel* on the two datasets.

Combining the results in Tables 5, 6 and 7, we confirm that the *word-combination kernel* can achieve better performance than the other two kernels on the Reuters-21578 dataset and 20 Newsgroups dataset. This demonstrates the effectiveness of our algorithm. We also

	n = 2	n = 3	n = 4
Reuters-21578			
micro-averaged F_1 score	0.9506	0.9416	0.9214
macro-averaged F_1 score	0.9071	0.8895	0.8484
20 Newsgroups			
micro-averaged F_1 score	0.8259 $\pm$ 0.009	0.8232 $\pm$ 0.009	0.8096 $\pm$ 0.010
macro-averaged F_1 score	0.8303 $\pm$ 0.008	0.8301 $\pm$ 0.009	0.8233 $\pm$ 0.009

Table 4. The impact of word combination length on performance. The *word-combination kernel*s are with lengths *n* = 2, 3 and 4 . The results for the 20 Newsgroups dataset are averaged over 10 runs of the algorithm

	WCK	WK	WSK
earn	**0.9823**	0.9693	0.9672
acq	**0.9521**	0.9394	0.9278
money	**0.8595**	0.8561	0.8167
grain	**0.9585**	0.9470	0.9549
crude	**0.9024**	0.8904	0.8966
trade	**0.9108**	0.8919	0.8981
interest	**0.8539**	0.7662	0.6904
ship	0.8205	0.8101	**0.8312**
micro-average	**0.9504**	0.9353	0.9276
macro-average	**0.9050**	0.8838	0.8729

Table 5. The F_1 scores of the *word-combination kernel* (WCK), *word kernel* (WK) and *word-sequence kernel* (WSK) for the eight categories of Reuters-21578 dataset. The *word-combination kernel* and *word-sequence kernel* are both with length n = 2

find that the *word-sequence kernel* performs better than the *word kernel* on the 20 Newsgroups dataset, while has a less performance on the Reuters-21578 dataset. It appears that the *word-sequence kernel* are more suitable for classifying long documents (After preprocessing, the average document length of the Reuters-21578 dataset is 66 words, while that of the 20 Newsgroups dataset is 118 words).

4.3.3 The comparison of computing efficiency

We compare the running time of the three kernels on a notebook with 2.40GHz Intel® Core™ Duo CPU and 2G Memory. Table 8 summarizes the test results. The results show that the preprocessing time accounts for only a small fraction of the total running time. The *word-combination kernel* takes a little more preprocessing time than the other two kernels because it needs an extra preprocessing of sentence boundary determination. From the results one can see that the training and test time of the *word-combination kernel* are more than, but roughly comparable with that of the *word kernel*. Yet the *word-sequence kernel* takes a far more running time than the other two kernels. It appears that the computing efficiency of the *word-combination kernel* and *word kernel* can meet the requirement of practical application, while the *word-sequence kernel* takes an excessively long running time even for the eight Reuters categories.

Among the three kernels, the *word kernel* has the lowest computational complexity, while the *word-sequence kernel* has the highest computational complexity. As can be seen from Table 8, the respective running time of the three kernels is consistent with their computational complexity. [2] proposed a dynamic programming formulation to speed up the calculation of the *word-sequence kernel*, but this kernel is still extremely computationally demanding and not computationally practical for large datasets.

5. Conclusions and Future Work

In this paper we propose a novel kernel called *word-combination kernel* for text classification. We aim to provide a text kernel which is practical and easy to use. We described this kernel and empirically tested it for text classification tasks on the Reuters-21578 dataset and 20 Newsgroups dataset. The performance of this kernel is compared to those of the classical *word kernel* and *word-sequence kernel.*

We devised the word combination features for this kernel. The word combination is a collection of different words co-occurring in the same sentence. The feature space of this kernel is generated by all word combinations of specified length. By restricting the words of a word combination to the same sentence and using multi-word combinations, the word combination features can capture similarity at a

	WCK	WK	WSK
alt.atheism	**0.7981$\pm$ 0.029**	0.7497$\pm$0.026	0.7799$\pm$0.037
comp.graphics	0.6992$\pm$0.029	0.6944$\pm$0.026	**0.7121 $\pm$ 0.026**
comp.os.ms-windows.misc	**0.7494 $\pm$ 0.030**	0.6950$\pm$0.022	0.7430$\pm$0.029
comp.sys.ibm.pc.hardware	**0.7138 $\pm$ 0.034**	0.6455$\pm$0.034	0.6936$\pm$0.013
comp.sys.mac.hardware	**0.8031 $\pm$ 0.021**	0.7553$\pm$0.030	0.7907$\pm$0.023
comp.windows.x	0.8154$\pm$0.019	0.7607$\pm$0.044	**0.8210 $\pm$ 0.031**
misc.forsale	**0.7792 $\pm$ 0.028**	0.7238$\pm$0.015	0.7515$\pm$0.038
rec.autos	**0.8767 $\pm$ 0.022**	0.8538$\pm$0.028	0.8602$\pm$0.021
rec.motorcycles	**0.9364 $\pm$ 0.018**	0.9162$\pm$0.021	0.9247$\pm$0.017
rec.sport.baseball	**0.9357 $\pm$ 0.017**	0.9040$\pm$0.018	0.9226$\pm$0.022
rec.sport.hockey	**0.9567 $\pm$ 0.017**	0.9285$\pm$0.017	0.9496$\pm$0.012
sci.crypt	0.8847$\pm$0.024	0.8622$\pm$0.019	**0.8943 $\pm$ 0.022**
sci.electronics	**0.7272 $\pm$ 0.033**	0.6938$\pm$0.034	0.7014$\pm$0.029
sci.med	0.8894$\pm$0.018	**0.8944 $\pm$ 0.022**	0.8901$\pm$0.015
sci.space	0.9020$\pm$0.016	0.8920$\pm$0.013	**0.9029 $\pm$ 0.017**
soc.religion.christian	**0.8143 $\pm$ 0.025**	0.7488$\pm$0.018	0.7800$\pm$0.017
talk.politics.guns	**0.8392 $\pm$ 0.013**	0.8013$\pm$0.032	0.8138$\pm$0.024
talk.politics.mideast	**0.9308 $\pm$ 0.009**	0.9075$\pm$0.020	0.9137$\pm$0.015
talk.politics.misc	**0.8188 $\pm$ 0.019**	0.7694$\pm$0.022	0.7805$\pm$0.024
talk.religion.misc	**0.6645 $\pm$ 0.040**	0.5456$\pm$0.030	0.5711$\pm$0.036
micro-average	**0.8259 $\pm$ 0.009**	0.7884$\pm$0.008	0.8110$\pm$0.009
macro-average	**0.8303 $\pm$ 0.008**	0.7939$\pm$0.007	0.8151$\pm$0.009

Table 6. The F_1 scores of the *word-combination kernel* (WCK), *word kernel* (WK) and *word-sequence kernel* (WSK) for the 20 Newsgroups dataset. The *word-combination kernel* and *word-sequence kernel* are both with length n = 2. The results for the 20 Newsgroups dataset are averaged over 10 runs of the algorithm

		P-value (S-test)
Reuters-21578		
WCK	**WK**	0.004 (>>)
WCK	**WSK**	0.04 (>)
WK	**WSK**	0.6 (~)
20 Newsgroups		
WCK	**WK**	0.00002 (>>)
WCK	**WSK**	0.02 (>)
WSK	**WK**	0.00002 (>>)

Table 7. The statistical significance test results using S-test between the *word-combination kernel* (WCK), *word kernel* (WK) and *word-sequence kernel* (WSK). ">>" indicates a strong evidence that the left-hand kernel is better than the right-hand one; ">" indicates a weak evidence that the left-hand kernel is better than the right-hand one; "~" indicates we can't decide which side is better

	WCK	WK (linear kernel)	WSK
Reuters-21578			
Preprocessing time	29s	25s	25s
Training time	418s	341s	25177s
Test time	133s	92s	11256s
Total	551s	458s	36458s
20 Newsgroups			
Preprocessing time	62s	53s	53s
Training time	558s	349s	36153s
Test time	367s	156s	19367s
Total	987s	538s	55573s

Table 8. The comparison of the running time among the *word-combination kernel* (WCK), *word kernel* (WK) and *word sequence kernel* (WSK). The running time is measured in seconds. The *word-combination kernel* and *word-sequence kernel* are both with length n = 2. The time for the 20 Newsgroups dataset is averaged over 10 runs of the classifiers

more specific level than single words. By discarding word order, the word combinations are more compatible with the flexibility of natural language and the feature dimensions of documents can be reduced significantly to obtain a more compact feature space.

We proposed a linear combination formulation to combine the *word-combination kernels* with different feature lengths.

We observed the impact of word combination length on performance. When feature length $n = 2$, the *word-combination kernel* can obtain the best performance. A computationally simple and efficient algorithm is proposed to calculate this kernel. The running time of this kernel is more than but roughly comparable to that of the *word kernel*. Experimental results show that the *word-combination kernel* achieves better performance than the *word kernel* and *word-sequence kernel* on the Reuters-21578 dataset and 20 Newsgroups dataset. It appears that this kernel can be an effective approach to the conventional text classification task.

The *word-combination kernel* can be used in conjunction with any kernel-based learning system. We will further research the use of this kernel to document clustering, ranking tasks, etc, and conduct more experiments on various types of text collections. We will also explore the feature selection method which can automatically select the significant word combination features and can be embedded into the *word-combination kernel*.

6. Acknowledgement

This work is supported by School of Automation Science and Electrical Engineering, Beijing University of Aeronautics & Astronautics.

References

[1] Allan, J. (2000). Natural language processing for information retrieval. *In:* NAACL/ANLP (tutorial notes).

[2] Cancedda, N., Gaussier, E., Goutte, C., Renders, J.M. (2003). Word-sequence kernels. *Journal of Machine Learning Research*, 3, p.1059–1082.

[3] Cancedda, N., Mahe, P. (2009). Factored sequence kernels, *Neurocomputing*, 72 (7-9) 1407-1413.

[4] Chang, C.C. and Lin, C.J. (2001). LIBSVM: a library for support vector machines. Software available at: http://www.csie.ntu.edu.tw/~cjlin/libsvm.

[5] Cristianini, N., Shawe-Taylor, J., Lodhi, H. (2002). Latent semantic kernels. *Journal of Intelligent Information Systems*, 18 (2-3) 127-152.

[6] Cristianini, N., Shawe-Taylor, J. (2004). *Kernel Methods for Pattern Analysis.* Cambridge University Press. Cambridge, UK.

[7] Dumais, S., Platt, J., Heckerman, D., Sahami, M. (1998). Inductive learning algorithms and representations for text categorization*, In:* Proceedings of ACM-CIKM'98.

[8] Furnkranz, J., Mitchell, T., Rilof, E. (1998). A case study in using linguistic phrases for text categorization on the www. *In:* Working Notes of the AAAI/ICML, Workshop on Learning for Text Categorization.

[9] Haussler, D. (1999). Convolution kernels on discrete structures. *In:* Technical Report UCS-CRL-99-10, UC Santa Cruz.

[10] Hickl, A., Williams, J., Bensley, J., Roberts, K., Shi, Y., Rink, B. (2006). Question answering with LCC's CHAUCER at TREC 2006. *In*: Proceedings of TREC'06.

[11] Joachims, T. (1998). Text categorization with Support Vector Machines: Learning with many relevant features. *In*: Proceedings of the 10th European conference on machine learning, p.137-142, Springer Verlag.

[12] Kehagias, A., Petridis, V., Kaburlasos, V.G., Fragkou, P. (2000). A comparison of word- and sense- based text categorization using several classification algorithms. *Journal of Intelligent Information Systems*, 21 (3) 227-247.

[13] Leopold, E., Kindermann, J. (2002). Text categorization with Support Vector Machines. How to represent texts in input space? *Machine Learning*, 46(1-3) 423-444.

[14] Lodhi, H., Saunders, C., Shawe-Taylor, J., Cristianini, N., Watkins, C. (2002). Text classification using string kernels. *Journal of Machine Learning Research*, 2, 419–444.

[15] Moschitti, A., Basili, R. (2004). Complex linguistic features for text classification: a comprehensive study. *In: Proceedings of ECIR-04*, p.181-196.

[16] Moschitti, A. (2008). Kernel methods, syntax and semantics for relational text categorization. *In:* Proceedings of CIKM, p.253– 262, New York, NY, USA.

[17] Özgür, L., Güngör, T. (2010). Text classification with the support of pruned dependency patterns. *Pattern Recogn. Lett.* 31 (12) 1598-1607.

[18] Rakotomamonjy, A., Bach, F., Canu, S., Grandvalet, Y. (2008). SimpleMKL. *Journal of Machine Learning Research*, 9, 2491-2521.

[19] Salton, G., McGill, M. (1983). *I*ntroduction to Modern Information Retrieval. McGraw-Hill, New York.

[20] Salton, G., Buckley, C. (1988). Term-weighting approaches in automatic text retrieval. *Information Processing and Management*, 24 (5) 513-523.

[21] Schölkopf , B., Smola, A. J. (2002). Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond. MIT Press.

[22] Sebastiani, F. (2002). Machine learning in automated text categorization. *ACM Computing Surveys*, 34(1)1-47.

[23] Siolas, G., d'Alché-Buc, F. (2000). Support Vector Machines based on a semantic kernel for text categorization. *In: IJCNN 2000*, Vol.5.

[24] Suzuki, J., Isozaki, H. (2006). Sequence and tree Kernels with statistical feature mining. *Advances in Neural Information Processing Systems*. 18, 1321-1328.

[25] Teo, C. H., Vishwanathan, S. V. N. (2006). Fast and space efficient string kernels using suffix arrays. *In: Proceedings of the 23th International Conference on Machine Learning*, ACM Press, Pittsburgh, p.929–936.

[26] Vishwanathan, S., Smola, A.J. (2004). Fast kernels for string and tree matching. Schölkopf, B., Tsuda, K., Vert, J.P., Editors, *Kernel Methods in Computational Biology*, MIT Press, p.113–130.

[27] Voorhees, E. M (2004). Overview of the TREC 2001 Question Answering track. *In: Proceedings of the Thirteenth Text Retrieval Conference* (TREC 2004).

[27] Voorhees, E. M. (2004). Overview of the TREC 2001 Question Answering track. *In: Proceedings of the Thirteenth Text Retrieval Conference* (TREC 2004).

[28] Wang, P., Domeniconi, C. (2008). Building semantic kernels for text classification using Wikipedia. *In: Proceeding of the14th ACM SIGKDD international conference on Knowledge discovery and data mining KDD 08*, p.713-721.

[29] Wittek, P., Sándor Darányi, Tan, C. L. (2009). Improving text classification by a sense spectrum approach to term expansion. *In: Proceedings of the Thirteenth Conference on Computational Natural Language Learning* (CoNLL '09), p.183-191.

[30] Yang, Y., Liu, X. (1999). A re-examination of text categorization methods. *In: Proceedings of the 22nd ACM SIGIR Conference on Research and Development in Information Retrieval*, p.42–49.

Author Biographics

Lujiang Zhang received the B.S. degree in computer science from Chinese Academy of Sciences. Currently, he is a PhD Candidate in the School of Automation Science and Electrical Engineering, Beijing University of Aeronautics & Astronautics, China. His research interests include machine learning, data mining, and information security.

Xiaohui Hu received the PhD degree from the School of Computer Science and Engineering, Beijing University of Aeronautics & Astronautics, China. Currently, he is a senior researcher and doctoral supervisor in the Institute of Software, Chinese Academy of Sciences. His research interests include information systems integration and computer simulation technology.

Shiyin Qin received the PhD degree from Zhejiang University, China. Currently, he is a professor and doctoral supervisor in the School of Automation Science and Electrical Engineering, Beijing University of Aeronautics & Astronautics. His research interests include Artificial Intelligence, pattern recognition, intelligent control, and complexity science.