I.J. Information Technology and Computer Science, **2010, 2, 40-46**
Published Online December 2010 in MECS (http://www.mecs-press.org/)

Using String Kernel for Document Clustering

Qingwei Shi[1,2]
1 Institute of Science & Technology Information of China, Beijing, China
2 School of Software Liaoning Technical University, Huludao, China
fxsteve@163.com

Xiaodong Qiao
Institute of Science & Technology Information of China, Beijing, China
qiaox@istic.ac.cn

Xu Guangquan
School of Computer Science & Technology Tianjin University, Tianjin, China
ganjue@tju.edu.cn

***Abstract*—In this paper, we present a string kernel based method for documents clustering. Documents are viewed as sequences of strings, and documents similarity is calculated by the kernel function. According to the documents similarity, spectral clustering algorithm is used to group documents. Experimental results shows that string kernel method outperform the standard k-means algorithm on the Reuters-21578 dataset.**

***Index Terms*—exploratory data analysis, document clustering, string kernel, spectral clustering, support vector machine**

I. Introduction

Data clustering is a fundamental problem in exploratory data analysis, with applications ranging from statistics, social sciences, biology or computer science. Clustering technique is used to identify groups of "similar behavior" in data sets and to discover the underlying structure hidden there, without any prior knowledge of the characteristics of the data [1]. Document (or text) clustering is a subset of data clustering, which borrows concepts from the fields of information retrieval (IR), artificial intelligence (AI) , natural language processing (NLP), and machine learning (ML),among others. Document clustering aims to group similar document and thus present an overview of class in a collection of documents.

A good clustering can be viewed as one that organizes a collection into groups such that the documents within each group are both similar to each other and dissimilar to those in other groups. The first challenge in a document clustering problem is to determine which features of a document are to be used to distinguish documents, in other words, to seek a document model to represent documents. Vector space model (VSM) [2] developed from IR community is today most commonly used to represent the topic of a document. In VSM, full set of documents are defined as corpus and the set of terms occurring in the corpus as dictionary. A document can be viewed as a bag of terms or bag-of-words. We can represent a bag as a vector in a space in which each dimension is associated with one term from the dictionary. Hence, a document is mapped into a space of dimensionality with the size of the dictionary, typically a very large number. Despite the high-dimensionality of the dictionary, the vector associated with a given document is sparse, i.e. has most of its entries equal to 0, since it will contain only very few of the vast number of possible words. Under the VSM model, a collection of n documents with m unique terms is represented as an $m \times n$ term-document matrix (where each document is a vector of m dimensions). The similarity between documents is calculated by the cosine of documents vector.

As mentioned above, VSM represents documents as a bag of words. A potential weakness of the vector model is that it encodes no information about word order. An alternative representation is to view documents as strings, and documents similarity is not only defined by occurrence of terms, but how frequently terms occur together, or in sequence. This motivates string kernel methods to deal with the problem of document clustering. Kernel methods or support vector machine [3] has been widely used in classification and clustering, which is to build a kernel function. A kernel function returns the inner product between the mapped data points in a higher dimensional space, in which linear algorithms could be used for clustering, classification and regression. In string kernels [4] [5], the features are not word frequencies or an implicit expansion thereof, but the extent to which all possible ordered subsequences of characters are represented in the document. At the same the time, spectral clustering method closed to Laplacian Eigen-maps could find a proper low dimensional representation of a data set in a high dimensional space. Spectral clustering algorithms often outperform traditional clustering algorithms and this make it feasible to clustering documents by combining the spectral clustering methods and string kernel.

In this paper, we use the string kernel to process documents as sequences of words. Documents similarity is calculated by the kernel function, and spectral clustering method is employed to clustering the documents according to the kernel matrix. To validate the performance of our method, we compare with the k-means clustering technique on the well know Reuters-21578 data set.

The rest of this paper is organized as follows. In section II, we introduce the string kernel model to represents the documents. Section III presents spectral clustering method to clustering documents according to the kernel matrix. The simulation setup and experimental results are presented in section IV. Finally, we conclude the paper in section V.

II. STRING KERNEL FOR DOCUMENTS

Kernels have been developed to compute the inner product between images of strings in high-dimensional feature spaces using dynamic programming techniques. The use of kernels on strings, and more generally on structured objects, makes it possible for kernel methods to operate in a domain that traditionally has belonged to syntactical pattern recognition, in this way providing a bridge between that field and statistical pattern analysis.

In this section we consider the problem of embedding two sequences in a high-dimensional space in such a way that their relative distance in that space reflects their similarity and that the inner product between their images can be computed efficiently. we describe a string kernel for documents clustering. Documents are treated as a sequence of words. Documents similarity is measured by the number of (possibly non-contiguous) matching subsequences shared by two documents. The assumption is that the more substrings documents contained in common, the more similar they are. Such substrings do not need to be contiguous, and the degree of contiguity of one such substring in a document determines how much weight it will have in the comparison In order to deal with non-contiguous substrings, it is necessary to introduce a decay factor, $\lambda \in (0, 1)$ (see (1) for more details) that can be used to weight the presence of a certain feature in a text.

We now describe the primary definition of Strings, alphabets and substrings for string kernel as bellows.

Definition 1 (Strings and alphabets) An alphabet Σ is a finite collection of symbols called characters. A string is a finite sequence $u=(u_1,u_2...,u_r)$ of characters from an alphabet Σ. The symbol Σ^* denotes the set of all strings of any length, i.e., $\Sigma^*=\cup_{i=1}^{\infty}\Sigma^i$. The number |u| of symbols in a string $u\in\Sigma^*$ is called the length of the string. Given two strings $u\in\Sigma^*$ and $v\in\Sigma^*$, the symbol $uv=(u_1,...,u_{|u|},v_1,...,v_{|v|})$ denotes the concatenation of the two strings.

Definition 2 (Subsequences and substrings) Given a string $u\in\Sigma^*$ and an index vector $i=(i_1,...,i_r)$ such that $1\le i_1\cdots\le i_r\le|u|$, we denote by *u[i]* the subsequence $(u_{i_1},...,u_{i_r})$. The index vector $(1,...,r)$ is abbreviated by *1: r*. Given two strings $v\in\Sigma^*$ and $u\in\Sigma^*$ where $|u|\ge|v|$ we define the index set $I_{u,v}=\{i:i(i+|v|-1)|i\in\{1,...,|u|-|v|+1\}\}$, i.e., the set of all consecutive sequences of length $|v|$ in $|u|$. Then the string v is said to be a substring of u if there exists an index vector $i\in I_{v,u}$ such that $v=u[i]$. The length $l(i)$ of an index vector is defined by $i_{|v|}-i_1+1$, i.e., the total extent of the subsequence (substring) v in the string u.

Definition 3 (All-subsequences kernel)

The feature space associated with the embedding of the all-subsequences kernel is indexed by $I = \Sigma^*$, with the embedding given by

$$\phi_u(s)=\left|\{i:u=s(i)\}\right|,u\in I$$

that is, the count of the number of times the indexing string u occurs as a subsequence in the string s. The associated kernel is defined by

$$k(s,t)=\langle\phi(s),\phi(t)\rangle=\sum_{u\in\Sigma^*}\phi_u(s)\phi_u(t)$$

An explicit computation of this feature map will be computationally infeasible even if we represent $\phi_u(s)$ by a list of its non-zero components, since if $/s/ = m$ we can expect of the order of

$$\min\left(\binom{m}{k},|\Sigma|^k\right)$$

distinct subsequences of length k. Hence, for $m > 2|\Sigma|$ the number of nonzeroentries considering only strings u of length $m/2$ is likely to be at least $|\Sigma|^{m/2}$, which is exponential in the length m of the string s.

All the kernels can be defined by an explicit embedding map from the space of all finite sequences over an alphabet Σ to a vector space F. The coordinates of F are indexed by a subset I of strings over Σ, that is by a subset of the input space. In some cases I will be the set Σ^p of strings of length p giving a vector space of dimension $|\Sigma|^p$, while in others it will be the infinite-dimensional space indexed by Σ^* .As usual we use ϕ to denote the feature mapping

$$\phi:s\mapsto(\phi(s))_{u\in I}\in F$$

For each embedding space F, there will be many different maps ϕ to choose from. For example, one possibility is to set the value of the coordinate $\phi_u(s)$ indexed by the string u to be a count of how many times u occurs as a contiguous substring in the input string s, while another possible choice is to count how many times u occurs as a (non-contiguous) subsequence. In this second case the weight of the count can also be tuned to reflect how many gaps there are in the different

occurrences. We can also count approximate matches appropriately weighting for the number of mismatches.

Kernel methods or support vector machine aim to build a kernel function. A kernel function returns the inner product between the mapped data points in a higher dimensional space, in which linear algorithms could be used for clustering, classification and regression. For the string case, string kernels are defined as a similarity measure between two sequences of characters *s* and *t*. The generic form of string kernels is given by

$$k_n(s,t)=\sum_{u\in\Sigma^n}\langle\phi_u(s)\cdot\phi_u(t)\rangle=\sum_{u\in\Sigma^n}\sum_{i:u=s[i]}\lambda^{l(i)}\sum_{j:u=t[j]}\lambda^{l(j)}$$
$$=\sum_{u\in\Sigma^n}\sum_{i:u=s[i]}\sum_{j:u=t[j]}\lambda^{l(i)+l(j)}$$

Where Σ^n represents the set of all non empty strings and λ is a weight or decay factor which can be chosen to be fixed for all substrings or can be set to a different value for each substring. A direct computation of these features would involve O($|\Sigma|^n$) time and space, and it can be reduced to O($n|s||t|$) depend on recursive algorithms presents as follow.

$$k_i'(s,t)=\sum_{u\in\Sigma^n}\sum_{i:u=s[i]}\sum_{j:u=t[j]}\lambda^{|s|+|t|-i_1-j_1+2}$$
$$i=1,2,...n-1$$

The Recursive computation method of the subsequence kernel is as follow.

$K_0^i(s,t)=1$ for all s,t,

$K_0^i(s,t)=0$ if min (|s|, |t|) < i,

$K_0(s,t)=0$ if min (|s|, |t|) < i,

$$K_0^1(sx,t)=\lambda K_i^1(s,t)+\sum_{j:t_j=x}K_{i-1}'(s,t[1:j-1])\lambda^{|t|-j+2}$$

$i=1,2,...n-1$

$$K_0(sx,t)=K_i(s,t)+\sum_{j:t_j=x}K_{i-1}'(s,t[1:j-1])\lambda^2$$

For the bias problem of documents length, we use (1) to normalize the feature vectors in the document feature space.

$$\hat{K}(s,t)=\langle\hat{\phi}(s)\cdot\hat{\phi}(t)\rangle=\left\langle\frac{\phi(s)}{\|\phi(s)\|}\cdot\frac{\phi(t)}{\|\phi(t)\|}\right\rangle$$
$$=\frac{1}{\|\phi(s)\|\|\phi(t)\|}\langle\phi(s)\cdot\phi(t)\rangle=\frac{K(s,t)}{\sqrt{K(s,s)K(t,t)}}\qquad(1)$$

We could compare with the similarity of documents by means of the string kernel described above and spectral clustering method can be used with the document similarity matrix.

III. Spectral Clustering for Documents

Spectral clustering [6] [7] strongly connects to the graph partitioning problem, which is based on eigende compositions of affinity, dissimilarity or kernel matrices. Let X = {x_1, . . . , x_n} be the set of patterns to cluster. Starting from X, we can build a complete, weighted undirected graph *G(V,A)* having a set of nodes V = {v_1, . . . , v_n} corresponding to the n patterns and edges defined through the n × n adjacency (also affinity) matrix A. The adjacency matrix for a weighted graph is given by the matrix whose element a_{ij} represents the weight of the edge connecting nodes i and j. Being an undirected graph, the property $a_{ij} = a_{ji}$ holds. Adjacency between two patterns can be defined as follows:

$$a_{ij}=\begin{cases}h(x_i,x_j) & \text{if i}\neq\text{j}\\ 0 & \text{otherwise}\end{cases}$$

The function h measures the similarity between patterns and typically a Gaussian function is used:

$$h(x_i,x_j)=\exp\left(-\frac{d(x_i,x_j)}{2\sigma^2}\right)$$

where *d* measures the dissimilarity between patterns and σ controls the rapidity of decay of h. This particular choice has the property that A has only some terms significantly different from 0, i.e., it is sparse.

The degree matrix D is the diagonal matrix whose elements are the degrees of the nodes of G.

$$d_{ii}=\sum_{j=1}^{n}a_{ij}$$

In this framework the clustering problem can be seen as a graph cut problem where one wants to separate a set of nodes S ⊂ V from the complementary set $\overline{S}=V\setminus S$. The graph cut problem can be formulated in several ways depending on the choice of the function to optimize. One of the most popular functions to optimize is the cut:

$$cut(S,\overline{S})=\sum_{v_i\in S,v_j\in\overline{S}}^{n}a_{ij}$$

It is easy to verify that the minimization of this objective function favors partitions containing isolated nodes. To achieve a better balance in the cardinality of S and $\overline{S}$ it is suggested to optimize the normalized cut

$$Ncut(S,\overline{S})=cut(S,\overline{S})\left(\frac{1}{assoc(S,V)}+\frac{1}{assoc(\overline{S},V)}\right)$$

where the association assoc(S, V) is also known as the volume of S:

$$assoc(S,V)=\sum_{v_i\in S,v_j\in\overline{S}}^{n}a_{ij}\equiv vol(S)=\sum_{v_i\in S}^{n}a_{ii}$$

The complexity in optimizing these objective functions is very high and for this reason it has been proposed to relax it by using spectral concepts of graph analysis. This relaxation can be formulated by introducing the Laplacian matrix:

$$L = D - A$$

which can be seen as a linear operator on G. In addition to this definition of Laplacian there are alternative definitions:

- Normalized Laplacian $L_N = D^{-1/2} L D^{-1/2}$
- Generalized Laplacian $L_G = D^{-1} L$
- Relaxed Laplacian $L_\rho = L - \rho D$

Each definition is justified by special properties desirable in a given context. The spectral decomposition of the Laplacian matrix can give useful information about the properties of the graph. In particular it can be seen that the second smallest eigenvalue of L is related to the graph cut and the corresponding eigenvector can cluster together similar patterns.

Spectral approach to clustering has a strong connection with Laplacian Eigenmaps. The dimensionality reduction problem aims to find a proper low dimensional representation of a data set in a high dimensional space. In each node in the graph, which represents a pattern, is connected just with nodes corresponding to neighboring patterns and the spectral decomposition of the Laplacian of the obtained graph permits to find a low dimensional representation of X. The authors point out the close connection with spectral clustering and Local Linear Embedding providing theoretical and experimental validations.

A. Shi and Malik algorithm

The algorithm proposed by Shi and Malik.[7] applies the concepts of spectral clustering to image segmentation problems. In this framework each node is a pixel and the definition of adjacency between them is suitable for image segmentation purposes. In particular, if x_i is the position of the i-th pixel and f_i a feature vector which takes into account several of its attributes, they define the adjacency as:

$$a_{ij} = \exp\left(-\frac{\left\| f_i - f_j \right\|^2}{2\sigma^2}\right) \cdot \begin{cases} \exp\left(-\frac{\left\| x_i - x_j \right\|^2}{2\sigma^2}\right) & \text{if } \left\| x_i - x_j \right\| < R \\ 0 & \text{otherwise} \end{cases}$$

Here R has an influence on how many neighboring pixels can be connected with a pixel, controlling the sparsely of the adjacency and Laplacian matrices. They provide a proof that the minimization of $Ncut(S, \overline{S})$ can be done solving the eigenvalue problem for the normalized Laplacian L_N. In summary, the algorithm is composed of these steps:

1. Construct the graph G starting from the data set X calculating the adjacency between patterns
2. Compute the degree matrix D
3. Construct the matrix $L_N = D^{-1/2} L D^{-1/2}$
4. Compute the eigenvector e_2 associated to the second smallest eigenvalue λ_2
5. $D^{-1/2} e_2$ Use segment G

In the ideal case of two non connected subgraphs, $D^{-1/2} e_2$ assumes just two values; this allows to cluster together the components of $D^{-1/2} e_2$ with the same value. In a real case the splitting point must be chosen to cluster the components of $D^{-1/2} e_2$ and the authors suggest to use the median value, zero or the value for which the clustering gives the minimum Ncut. The successive partitioning can be made recursively on the obtained sub-graphs or it is possible to use more than one eigenvector.

B. Ng, Jordan and Weiss algorithm

The algorithm that has been proposed by Ng et al. [23] uses the adjacency matrix A as Laplacian. This definition allows to consider the eigenvector associated with the largest eigenvalues as the "good" one for clustering. This has a computational advantage since the principal eigenvectors can be computed for sparse matrices efficiently using the power iteration technique. The idea is the same as in other spectral clustering methods, i.e., one finds a new representation of patterns on the first k eigenvectors of the Laplacian of the graph.

The algorithm is composed of these steps:

1. Compute the affinity matrix $A \in \mathbb{R}^{n \times n}$:

$$a_{ij} = \begin{cases} h(x_i, x_j) & \text{if i} \neq \text{j} \\ 0 & \text{otherwise} \end{cases}$$

2. Construct the matrix D
3. Compute a normalized version of A, defining this Laplacian:

$$L = D^{-1/2} A D^{-1/2}$$

4. Find the k eigenvectors $\{e_1, \ldots, e_k\}$ of L associated to the largest eigenvalues $\{\lambda_1, \ldots, \lambda_k\}$.
5. Form the matrix Z by stacking the k eigenvectors in columns.
6. Compute the matrix Y by normalizing each of the Z's rows to have unit length:

$$y_{ij} = \frac{z_{ij}}{\sum_{r=1}^{k} z_{ir}^2}$$

In this way all the original points are mapped into a unit hyper sphere.

7. In this new representation of the original n points apply a clustering algorithm that attempts to minimize distortion such as K-means.

As a criterion to choose σ they suggest to use the value that guarantees the minimum distortion when the clustering stage is performed on Y . They tested this algorithm on artificial data sets showing the capability of the algorithm to separate nonlinear structures.

In our case we use a string kernel to define the similarity between two documents and construct the kernel matrix. The data is then embedded into the subspace of the largest eigenvectors of the normalized kernel matrix. Supposed that the similarity of two documents is $s_{ij} = s(x_i, x_j)$, the spectral clustering algorithm is as follows.

Input: Similarity matrix $S \in \mathbb{R}^{n \times n}$, number k of clusters to construct.

1. Compute the diagonal degree matrix D with elements:

$$d_i = \sum_{j=1}^{n} s_{ij}$$

2. Compute the unnormalized Laplacian L.

$$L = D - S$$

3. Compute the first k eigenvectors $u_1, u_2, \ldots, u_k$ of L.

4. Let $U \in \mathbb{R}^{n \times k}$ be the matrix containing the vectors $u_1, u_2, \ldots, u_k$ as columns.

5. For $i = 1, \ldots, n$, let $y_i \in \mathbb{R}^k$ be the vector corresponding to the i-th row of U

6. Cluster the points $(y_i)_{i=1,\ldots,n}$ in R_k with the k-means algorithm into clusters $C_1, \ldots, C_k$.

Output: Clusters $A_1, \ldots, A_k$,with $A_i = \{ j | y_j \in C_i \}$.

IV. SIMULATION RESULTS

A. Data Set

The Reuters-21578 dataset [8] contains stories for the Reuters news agency. It is currently one of the most widely used datasets for text categorization research. In our experiments we used a subset of the Reuters dataset so that the computation of a full kernel matrix in memory was not a concern. We used the "crude" which contains about 580 documents, the "corn" category which includes 280 documents and a sample of 1100 documents from the "acq" category. Our dataset thus consists of 1720 documents after preprocessing: We removed the stop words that occur in a stop list and any empty documents and convert all characters to lower case. We also removed punctuation and white space and performed stemming on the documents.

B. Experimental Setup

We perform clustering on the dataset using the k-means and spectral clustering methods with the kernlab package in R. In order to learn more about the effect of the string kernels hyper-parameters on the clustering results we run the clustering algorithms over a range of the length parameter n which controls the length of the strings compared in the two character sets and the decay factor λ. We study the effects of the parameters by keeping the value of the decay parameter λ fixed and varying the length parameter. Note that for each parameter set a new kernel matrix containing different information has to be computed. We use values from n = 3 to n = 14 for the length parameter and λ = 0.2, λ = 0.5 and λ = 0.8 for the decay factor.

We also use both the string (or spectral) and the full string kernel and normalize in order to remove any bias introduced by document length. We thus use a new embedding $\hat{\phi} = \frac{\phi(s)}{\|\phi(s)\|}$ which gives rise to the kernel:

$$\hat{K}(s,s') = \left\langle \hat{\phi}(s) \cdot \hat{\phi}(s') \right\rangle = \left\langle \frac{\phi(s)}{\|\phi(s)\|} \cdot \frac{\phi(s')}{\|\phi(s')\|} \right\rangle$$

$$= \frac{1}{\|\phi(s)\| \|\phi(s')\|} \left\langle \phi(s) \cdot \phi(s') \right\rangle = \frac{K(s,s')}{\sqrt{K(s,s)K(s',s')}}$$

For the classical k-means method we create a term document matrix of the term frequencies and also an inverse term frequencies matrix.

C. Performance measure

We evaluate the performance of the various clustering techniques using the recall rate which is a typical measure for evaluating the performance of a clustering algorithm when the actual labels of the clustered data are known. Given a discovered cluster γ and the associated reference cluster Γ, recall R is defined as:

$$R = \frac{\sum_{\Gamma=1}^{k} n_{\gamma}^{\Gamma}}{\sum_{\Gamma=1}^{k} n_{\Gamma}}$$

where n_{γ}^{Γ} is the number of documents from reference cluster Γ assigned to cluster γ, n_{Γ} is the total number of documents in cluster Γ.

D. Results

The main goal of these experiments is to establish if kernel methods along with string kernels are an available solution for grouping a set of text documents. From the experiments we run it became obvious that the λ parameter influences the performance only minimally and thus we chose to look at the results in relation to the string length kernel parameter which seems to have a more profound influence on the performance of the kernel-based clustering methods. The performance of the k-means clustering method is also very similar with both the simple document matrix and the inverse frequency document matrix.

Fig. 1 shows the average recall rate over 10 runs for the spectral clustering methods, and the k-means method with the full string kernel compared to the reference recall rate of the inverse term document matrix clustered with a simple k-means algorithm. The plot shows that the spectral method fail to improve over the performance of the standard k-means clustering technique. We also note that the spectral clustering technique provides very stable results. This can be attributed to the fact that the projection of the data into the eigenspace groups the data into tight clusters which are easy to separate with a standard clustering technique.

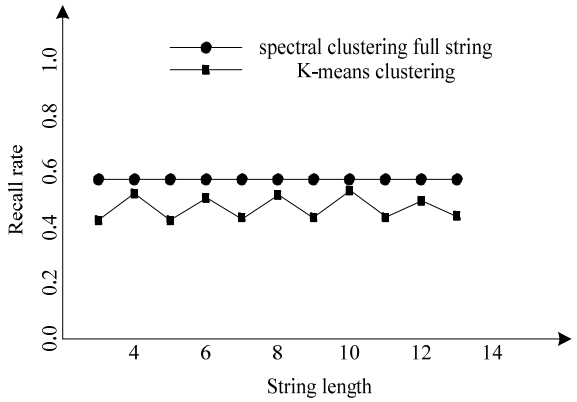

Figure 1. Note how the caption is centered in the column.

Fig. 2 shows the recall rate of the spectral clustering method with a string kernel averaged over 10 runs compared to the standard k-means clustering results. This is clearly the best performing clustering method for this set of text documents and also exhibits some interesting behavior. For rather small lengths of substrings considered (3, 4, 5) the performance seems to increase monotonically and at the value of 6 hits a threshold. For the range of values between 6 and 10 the performance increase is much smaller and for the value of 10 the highest recall rate of 0.927 is reached. For higher values of the length parameter the performance drops sharply only to increase again for a string length value of 14. Again this method is very stable and exhibits minimal variance.

We also evaluated the methods in terms of running time. The experiments run on a Linux machine with a 2.6

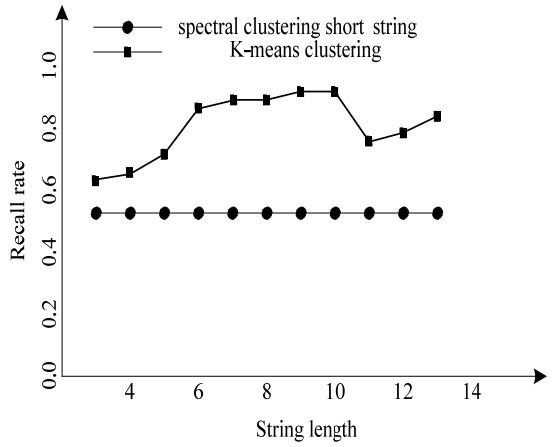

Figure 2. Note how the caption is centered in the column.

GHz Pentium 4 CPU. Table 1 provides the running time for the calculation of a full kernel matrix and the running time for the clustering methods. Note that the running time for the kernel-based clustering methods is the time needed to cluster data with a precomputed kernel matrix. From the results it is clear that most of the computing time is spend on the calculation of the kernel matrix.

TABLE I. TIMIE FOR CLUSTER METHODS

Methods	***Time***
Kernel matrix calculations	2h
spectral clustering	20 sec
k-means	40 sec

V. CONCLUSIONS

In this paper, we present a document clustering methods combined with the string kernel and spectral clustering algorithm. Experiments on the Reuters-21578 dataset show that this method outperforms the standard k-means method in recall rate with a proper string length.

One drawback of the kernel based methods is the time on which spend the computation of the kernel matrix. A suffix tree based implementation of the string kernels as in [11][12] [18] could provide a solution to this issues.

ACKNOWLEDGMENT

This work was supported by the foundation of department of education of liaoning province under grant No. L2010168.

REFERENCES

[1] A.K. Jain, M.N. Murty, and P.J. Flynn, *Data clustering: a review*, ACM computing surveys, vol.31, September 1999, pp. 264-323.

[2] G. Salton, A. Wong, and C. S. Yang, *A vector space model for automatic indexing,* Communications of the ACM, vol.18, November 1975, pp.613-620.

[3] I. S. Jacobs and C. P. Bean, *Fine particles, thin films and exchange anisotropy,* in *Magnetism*, vol. III, G. T. Rado and H. Suhl, Eds. New York: Academic, 1963, pp. 271–350.

[4] N. Cristianini and J. Shawe-Taylor, *An introduction to Support Vector Machines,* Cambridge University Press, Cambridge, UK, 2000.

[5] H. Lodhi, N. Cristianini, J. Shawe-Taylor, and C. Watkins, *Text classication using string kernel,* The Journal of Machine Learning Research, vol.2, MIT Press Cambridge, MA, USA, 2001, pp.419-444.

[6] J. Shawe-Taylor and N. Cristianini, *Kernel methods for pattern analysis,* Cambridge University Press, Cambridge, UK, 2004.

[7] J. Shi and J. Malik, *Normalized cuts and image segmentation,* Transactions on Pattern Analysis and Machine Intelligence, vol.22, 2000, pp.888–905.

[8] U. von Luxburg, *A tutorial on spectral clustering,* Statistics and Computing, vol.17, Springer, 2007, pp.395-416.

[9] David Lewis. Reuters-21578 text categorization test collection, 1997

[10] Charles Fowlkes, Serge Belongie, Fan Chung, and Jitendra Malik, *Spectral grouping using the nystrom method,* Transactions on Pattern Analysis and Machine Intelligence, vol.26(2), 2004, pp.214-225.

[11] S.V.N. Vishwanathan and A.J. Smola, *Fast kernels for string and tree matching,* Kernels and Bioinformatics, Cambridge, MA, MIT Press, 2004.

[12] K. I. Christoper, Williams and Matthias Seeger, *Using the Nystrom method to speed up kernel machines,* Advances in Neural Information Processing Systems 13, Cambridge, MA, MIT Press, 2001, pp. 682– 688.

[13] Charles Elkan, *Using the triangle inequality to accelerate k-means,* In Proceedings of the Twentieth International Conference on Machine Learning(ICML'03), 2003, pp. 147-153.

[14] M. Filippone, F. Camastra, and F. Masulli and S. Rovetta , *A survey of kernel and spectral methods for clustering,* Pattern Recognition, vol.41(1), Elsevier, 2008, pp. 176-190.

[15] Nicholas O. Andrews and Edward A. Fox, *Recent Developments in Document Clustering,* Technical Report TR-07-35, Computer Science, Virginia Tech, 2007.

[16] D. Yan, L. Huang and M.I. Jordan, *Fast approximate spectral clustering,* In Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining, ACM Press, 2009, pp. 907-916.

[17] J. Shawe-Taylor and N. Cristianini, *Kernel methods for pattern analysis,* Cambridge University Press, 2004.

[18] A. Karatzoglou and I. Feinerer, *Kernel-based machine learning for fast text mining in R,* Computational Statistics \& Data Analysis, vol. 54(2) Elsevier, pp. 290-297.

[19] A. Moschitti, *Syntactic and semantic kernels for short text pair categorization,* In Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics, 2009, pp. 576-584.

[20] O. Amayri and N. Bouguila, *Improved Online Support Vector Machines Spam Filtering Using String Kernels,* Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications, springer, 2009, pp. 621-628.

[21] S. Zheng, Y.Yang, H. Wu and W. Liu, *Chinese Text Classification Using Key Characters String Kernel,* Fifth International Conference on Semantics, Knowledge and Grid, 2009, pp.113-119.

[22] Ralf Herbrich, *Learning Kernel Classifiers Theory and Algorithms,* Adaptive Computation and Machine Learning, MIT Press, 2002.

[23] A. Y. Ng, M. I. Jordan, and Y. Weiss. *On spectral clustering: Analysis and an algorithm*, In T. G. Dietterich, S. Becker, and Z. Ghahramani, editors, Advances in Neural Information Processing Systems 14, Cambridge, MA, MIT Press, 2002.

Qingwei Shi was born in fuxin,China in 1973, earned B.S. degree in the field of management from Dalian Maritime University in 1995 and M.S. degree in the field of computer science & technology from Shenyang University of Technology in 2001, earned Ph.D. degree in the field of computer science & technology from Tianjin University in 2008.

He is now a teacher at school of software in Liaoning Technical University(LNTU). Before joining the LNTU, he was a software engineer in GKT corporation for about 3 years. Recently he has published 10 researh papers in the area of text mining.

