Topic Sequence Kernel

Jian Xu, Qin Lu, Zhengzhong Liu, and Junyi Chai

Department of Computing, The Hong Kong Polytechnic University
{csjxu,csluqin,hector.liu,csjchai}@comp.polyu.edu.hk

Abstract. This paper addresses the problem of classifying documents using the kernel approaches based on topic sequences. Previously, the string kernel uses the ordered subsequence of characters as features and the word sequence kernel is proposed to use words as the subsequences. However, they both face the problem of computational complexity because of the large amount of symbols (characters or words). This paper, therefore, proposes to use sequences of topics rather than characters or words to reduce the number of symbols, thus increasing the computational efficiency. Documents that exhibit similar posterior topic proportions are expected to have similar topic sequence and then should be classified into the same category. Experiments conducted on the Reuters-21578 datasets have proven this hypothesis.

Keywords: Topic sequence, string kernel, classification.

1 Introduction

The Support Vector Machine (SVM) has been widely applied in [1-3] and it is well known for using kernel methods to handle non-separable data points by the hyperplane in the kernel space. The commonly used kernels are linear, polynomial, and RBF kernels. [4] proposed the string kernel which took the ordered subsequence of characters for document representation. This kernel considers the sequential order between characters in a document. However, measuring similarity between two sequences requires a lot of computational resources. To resolve this problem, [4] proposed a dynamic programming technique to promote computation efficiency. To further reduce the computational complexity, [5] used the words instead of characters as the sequences for document representation.

In this paper, we extend the basic idea of string kernel to represent documents as sequences of topics instead of words or characters. As topics are a summary of documents, they can better capture the document semantics. One document might be about crude oil (0.6), ship (0.3) and trade (0.1). Another document may be about trade (0.5), crude oil (0.3) and ship (0.2). Two documents both have three topics but with different topic proportions. For the first document, crude oil has the greatest proportion among the three topics, ship the second greatest proportion and trade is the least. Intuitively, most important topic will be first expressed, and less important topic will be conveyed in succession and the least important topic will be delivered in the last. In so doing, a document can be represented in a sequence of topics according to the topic

Y. Hou et al. (Eds.): AIRS 2012, LNCS 7675, pp. 457–466, 2012.

proportion. It is reasonable to assume that if two documents are similar, they are expected to have not only similar topics, but also the topic sequences. Based on this assumption, we try to classify texts based on the kernel approach using the sequence of topics instead of words or characters. This could greatly reduce the computational complexity of string kernel as there is a small number of topics compared to the words in the whole document collection.

The rest of this paper is organized as follows. Section 2 describes the related works of kernel methods and topic modeling approaches. Section 3 presents the approach of generating topic sequences using topic modeling technique and introduces the string kernel using the topic sequences. Section 4 gives the performance evaluation. Section 5 is the conclusion.

2 Related Works

Kernel functions are computational shortcuts that are able to represent linear patterns in high-dimensional space [6]. They are used to compute pairwise inner products between mapping examples in the feature space [4]. A kernel is valid only when it meets the Mercer's conditions: symmetry and positive semi-definiteness [7].

Currently, there are various kinds of kernels used in SVM, including the polynomial kernel, radio basis function (RBF), and so on. Different from previous kernels that are dependent on the word frequencies, the string kernel takes into account the relative positional information of characters in documents. It compares two documents by enumerating the substrings they contain: the more substring they share, the more similar they are [4].

In this paper, we extend the basic idea of string kernel using the sequence of topics. Hence, topic modeling is vital to generate topic sequences. Methods to discover the semantic structure of a document collection using the probabilistic model include latent semantic indexing (LSI), probabilistic LSI, latent Direchlet allocation (LDA) [8-10]. Besides, two extensions to LDA has been proposed: the correlated topic model and the dynamic topic model [11-12]. To find the posterior distribution of the latent topics given the document collection, various approximate approaches have been used, including mean-field variational inference [8], expectation propagation [13] and Gibbs sampling [14] and collapsed variational inference [15].

3 Methodology

3.1 Topic Modeling and Sequence Generating

Documents that have similar topics are expected to exhibit similar topic proportions. Important topics will have large proportions in a document. Suppose there are four documents: ***Doc1***, ***Doc2***, ***Doc3*** and ***Doc4***, and three topics: *trade*, *crude oil*, and *ship*. The topic proportions in these four documents are:

Doc1: *trade (0.3), crude oil (0.5), ship (0.2)* ***Doc2:*** *trade (0.1), crude oil (0.3), ship (0.6)*
Doc3: *trade (0.3), crude Oil (0.6), ship (0.1)* ***Doc4:*** *trade (0.2), crude oil (0.3), ship (0.5)*

Obviously, ***Doc1*** and ***Doc3*** have similar topic proportions. They both have *crude oil* as the most important topic, *trade* as the secondary important topic and *ship* as the least important topic. Similarly, ***Doc2*** and ***Doc4*** have a sequential order of *ship*, *crude oil* and *trade* according to the topic proportions. In this sense, ***Doc1*** is similar to ***Doc3*** and ***Doc2*** to ***Doc4***. Besides, ***Doc1*** and ***Doc3*** are similar because they have the same sequential topic order. Therefore, the topics in four documents can be re-arranged in a sequential order according to their proportions in each document.

Doc1: *crude oil (0.5), trade (0.3), ship (0.2)* ***Doc2:*** *ship (0.6), crude oil (0.3), trade (0.1)*
Doc3: *crude oil (0.6), trade (0.3), ship (0.1)* ***Doc4:*** *ship (0.5), crude oil (0.3), trade (0.2)*

Next is to obtain these topic distributions. In this paper, topics are modeled through the LDA [10]. LDA is a generative model which is based on probabilistic sampling techniques investigating how words in documents are generated with the hidden variables [14]. Its main idea is to model documents in terms of topics where a topic is defined as a distribution over a fixed vocabulary of words. In this model, words in documents are observable and topics are latent variables hidden in these documents. Its graphical representation is given in Fig. 1.

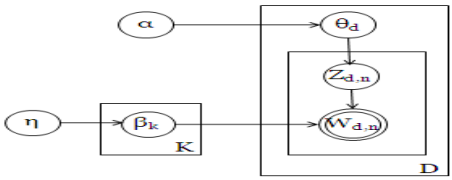

Fig. 1. LDA Graphical Representation

In the Fig.1, each node denotes a random variable and the edge between nodes represents dependency relations between nodes. The double circles around the random variable denote an observable node (evidence node). The plate surrounding the nodes indicates N i.i.d samples. D and K refer to the number of documents and the number of topics, respectively. α and η are hyper-parameters on the mixture proportions for topics and documents. θ_d refers to the multinomial topic distributions for document d and β_k is multinomial word distributions for topic k. $Z_{d,n}$ denotes a topic from which the n^{th} word in document d is drawn and $W_{d,n}$ indicates the observable n^{th} word in d.

In LDA model, for a document d, a vector of topic distributions $\vec{\theta}_d$ is drawn from a Dirchlet distribution; topic assignment for n^{th} word $Z_{d,n}$ follows from a multinomial distribution; and the n^{th} word $W_{d,n}$ in document d is sampled from multinomial distribution. To generate topic sequences, $p(z|w)$ must be obtained for the hyper-parameters α and η. Since exact inference of this distribution is intractable [10], Gibbs sampler is used. When $p(z|w)$ is obtained, the topic distributions θ for each document can be estimated. These topic distributions are used to generate topic sequences.

3.2 Topic Sequence Kernel

The topic sequences are generated in Section 3.1. Then, the subsequences of the topic sequences are extracted as features in the topic sequence kernel model.

Given Σ as a finite topic set, let $S=z_1z_2\ldots z_{|S|}$ be a sequence of topics for a document, $z_i \in \Sigma$ and $1 \le i \le |S|$. A subsequence of *S,* denoted by *u,* the feature used in the string kernel model, is defined by a index sequence $I=(i_1,\ldots,i_n)$ of *S* such that $1 \le i_1 < i_2 < i_n \le |S|$ *and* $u=S[I]$, where *n* is the length of *u*, the number of topics of the subsequence *u*. The span of *S[I]* , denoted by *l(I),* is the distance of the first topic and the last topic of *u* in *S*, calculated by $i_n - i_1 + 1$. For example, if *S* is the topic sequence of $z_1z_2z_4z_3$ and $u = z_1z_4$, then the index set, $I=[1,3]$ such that $u=S[1,3]$, and the span of $S[1,3]$ is 3-1+1 =3. The feature matching of *u* for a given topic sequence *S,* denoted by ϕ, is:

$$\phi_u(S) = \sum_{I:u=S[I]} \lambda^{l(I)}$$

where λ is the decay factor, in the range of [0,1], that penalizes the longer span *l(I)* of subsequences. Based on topic sequence, any two documents, represented by their topic sequences *S,* and *T*, are compared through the topic subsequences as features. To control the feature space, the topic sequence kernel has a parameter *n* which denotes the length of subsequences in the feature space. Then, the similarities are:

$$K_n(S,T) = \sum_{u \in \sum^n} \langle \phi_u(S) \cdot \phi_u(T) \rangle = \sum_{u \in \sum^n} \sum_{I:u=S[I]} \lambda^{l(I)} \sum_{J:u=T[J]} \lambda^{l(J)}$$

$$= \sum_{u \in \sum^n} \sum_{I:u=S[I]} \sum_{J:u=T[J]} \lambda^{l(I)+l(J)}$$

where $\sum^n$ is the set of all topic subsequences of length *n*. *S[I]* and *T[J]* are the subsequences in *S* and *T*. *l(I)* and *l(J)* are the spans of the subsequences in *S* and *T*.

In fact, each topic sequence has the unique topics. This means that the subsequences will occur only once in a topic sequence. Therefore, the feature matching of u in the topic sequences *S* and *T* will be changed to,

$$\phi_u(S) = \lambda^{l(I)} \quad \text{and} \quad \phi_u(T) = \lambda^{l(J)}$$

And the kernel function $K_n(S,T)$ will be changed to,

$$K_n(S,T) = \sum_{u \in \sum^n} \langle \phi_u(S) \cdot \phi_u(T) \rangle = \sum_{u \in \sum^n} \sum_{I:u=S[I]} \lambda^{l(I)} \sum_{J:u=T[J]} \lambda^{l(J)} = \sum_{u \in \sum^n} \lambda^{l(I)+l(J)}$$

In this sense, the computational cost will be reduced due to the cancellation of summation procedure in each topic sequence. To avoid enumeration of all subsequences for similarity measurement, dynamic programming, similar to the method in [4] is used here for similarity calculation.

4 Performance Evaluation

Experiments are conducted on the Reuters-21578 dataset, from which we used Modified Apte ("ModeApte") split. Due to the concern of computational complexity of the string kernel, [4] drew a subset of 470 documents with 380 documents for training and 90 documents for testing. [5] proposed to used word sequence kernel on the ten frequent categories. This word sequence kernel, however, is still resource demanding as they claimed. In this paper, the topic sequence kernel will not suffer from this problem since the number of topics is much less than that of characters or words.

In the following experiments, the values for the hyper-parameters α and β are 50/K and 0.01 [14] and the number of iterations is set to 500. Note that the training and testing documents are placed together to obtain the posterior topic distribution. Based on topic distributions in each document, a topic sequence is created for the document.

In terms of classifier, the LIBSVM[1] tool with the one-versus-one strategy is used and default parameters are kept. Since the training sets for the ten categories are unbalanced in favor of negative examples, we weigh the relative importance of positive and negative examples by the ratio between negative and positive examples [5].

For evaluation, we used the precision (**p**), recall (**r**) and F-measure (**F**). In order to have a general overview of performance on the ten categories, the micro-averaged and macro-averaged performances are used. To control the effectiveness of the topic sequence kernel, three parameters need to be tuned manually: length of a subsequence l and the decaying factor λ and the number of topics K. Table 1 gives the overall best performance when $\lambda=0.55$ and $l=2$ and $K=10$. In the following experiments, ***TSK*** is used to denote the topic sequence kernel.

Table 1. Best Performance when λ=0.55, $l=2$ and $K=10$

	Micro-average			Macro-average		
	p	**r**	**F**	**p**	**r**	**F**
TSK	87.06	86.69	86.88	76.48	75.91	75.98

Table 1 shows that the best performance is achieved when the number of topics K is 10. And $K=10$ is the number of categories in this experimental dataset. This means that if the number of topics is known beforehand, the LDA model can well capture the topic structure in documents. The detailed classification results for each category are listed in Table 2.

From the Table 2, we found that the *earn* category gives the best performance because of its lowest negative-to-positive ratio and the *acq* category gives the second best performance. It is interesting to note that although the *grain* category has a higher negative-to-positive ratio than the *money-fx* category does, it attained a better performance than the *money-fx* category. However, higher negative-to-positive ratio will naturally produce worse classification results. The *corn* and *ship* categories have

[1] http://www.csie.ntu.edu.tw/~cjlin/libsvm/

Table 2. Detailed Performance of each Category when λ=0.55, *l=2* and *K=10*

Category	p	r	F
corn	58.93	58.93	58.93
ship	73.91	76.40	75.14
wheat	66.67	78.87	72.26
acq	**93.28**	**92.63**	**92.95**
crude	78.98	73.54	76.16
earn	**95.07**	**95.77**	**95.42**
grain	88.89	85.91	87.37
interest	64.08	50.38	56.41
money-fx	66.18	76.54	70.98
trade	78.85	70.09	74.21

well illustrated this. In the following sections, we will study the effectiveness of the topic sequence kernel (***TSK***) by varying the length of a subsequence *l*, the decaying factor λ and the number of topics *K*.

4.1 Effectiveness of Varying the Number of Topics

The number of topics is crucial to the topic sequence kernel. It determines the computational complexity of the ***TSK***. Therefore, for this set of experiments, we kept the values of the parameters the subsequence length *l=2* and the decaying factor λ*=0.55* fixed and observed how the performance is influenced by the number of topics *K* from 5 to 20.

Table 3. Performance of Varying the Topic Number when λ*=0.55* and *l=2*

	Micro-average			Macro-average		
K	**p**	**r**	**F**	**p**	**r**	**F**
5	84.62	73.23	78.52	40.2	42.13	41.14
10	**87.06**	**86.69**	**86.88**	**76.48**	**75.91**	**75.98**
15	84.76	84.82	84.79	70.43	71.01	70.24
20	82.37	82.96	82.66	64.67	66.47	65.47

Table 3 shows that when the topic number *K* is 10, the system gives the best performance, implying that latent topics in the document collection are well captured by *K=10* and other configurations cannot detect the topic structures properly if the *K* is too large or too small. Moreover, *K*=10 is the number of categories of document collections. On the other hand, if the number of topics is known beforehand, the latent topics can be well modeled out of the document collections by LDA. It is worth noting the case of *K=5* in which the micro-average score is high, but the

macro-average score is rather low. This is because the *corn*, *ship* and *interest* categories all get zero classification precision and recall values. However, the *acq* and *earn* categories get high precision and recall and these two categories have a large number of testing documents, thus contributing to the overall higher micro-average score.

4.2 Effectiveness of Varying the Subsequence Length

In this set of experiments, the values of the number of topics $K=10$ and the decaying factor $\lambda=0.55$ are fixed and we analyze the effect of varying the subsequence length from 2 to 6.

Table 4. Performance of Varying Subsequence Length when $\lambda=0.55$ and $K=10$

	Micro-average			Macro-average		
l	**p**	**r**	**F**	**p**	**r**	**F**
2	**87.06**	**86.69**	**86.88**	**76.48**	**75.91**	**75.98**
3	74.93	78.4	76.63	58.75	63.83	61
4	78.14	75.53	76.81	63.89	57.45	60.24
5	81.7	73.52	77.4	65	50.34	55.81
6	84.03	66.81	74.44	66.33	38.47	45.65

Table 4 shows that the ***TSK*** can be more effective for smaller subsequence as compared to larger subsequences since the smaller topic subsequences are able to capture the document semantics than the longer ones. Besides, the longer subsequences have a strict requirement over the matching unit of the two sequences. For example, the topic sequence $S=z_1z_2z_3z_4$ and $T=z_2z_1z_4z_3$, if the subsequence length is set to 3, we will have a set of subsequences $\{z_1z_2z_3, z_1z_3z_4, z_2z_3z_4\}$ from S and another set of subsequences $\{z_2z_1z_3, z_2z_1z_4, z_1z_4z_3\}$ from T. Clearly, we will find no intersections between the two subsequence sets. If the subsequence length is set to 2, we will find an intersection set $\{z_2z_3, z_2z_4, z_1z_3, z_1z_4\}$ between S and T. Hence, we obtained the best micro-average and macro-average scores when the subsequence length is set to 2.

4.3 Effectiveness of Varying the Decaying Factor

The decaying factor λ controls how many gaps are allowed in the matching subsequences of the two sequences. If $\lambda=1$, the gaps between the subsequences are not penalized. If $0 < \lambda < 1$, the larger the gaps are, the more penalty will be placed on the subsequence. For this set of experiments, we kept $K=10$ and $l=2$ fixed and studied the effects of varying the decaying factor λ.

In Table 5, the highest micro-precision is achieved at $\lambda= 0.1$ while all other highest micro-average and macro-average scores are obtained at $\lambda=0.55$. The higher values of λ will place more weights to contiguous topic subsequences. In other words, this is the parameter that penalizes the topic subsequences with large interior gaps.

Table 5. Performance of Varying Decaying Factor when *l=2* and *K=10*

	Micro-average			Macro-average		
λ	**p**	**r**	**F**	**p**	**r**	**F**
0.1	**87.91**	82.45	85.1	74.01	66.1	67.56
0.15	86.03	82.85	84.41	74.61	72.06	72.87
0.2	86.33	80.7	83.42	65.09	61.96	63.31
0.25	85.85	82.92	84.36	73.15	70.53	71.39
0.3	87.22	85.22	86.21	74.1	72.5	72.96
0.35	87.84	84.54	86.16	72.89	66.8	68.4
0.4	86.28	83.96	85.11	73.62	70.05	70.94
0.45	85.52	81.81	83.62	72.97	65.41	68.49
0.5	86.18	85.04	85.61	75.1	73.51	73.71
0.55	87.06	**86.69**	**86.88**	**76.48**	**75.91**	**75.98**
0.6	84.28	80.8	82.51	69.47	62.88	64.51
0.65	84.86	81.66	83.23	71.38	64.96	66.86
0.7	85.84	83.49	84.65	70.73	64.7	66.49
0.75	84.55	86	85.27	71.57	72.17	70.67
0.8	84.65	85.11	84.88	72.08	72.43	71.93
0.85	84.92	84.64	84.78	72.16	69.49	69.79
0.9	85.37	85	85.19	71.76	70.34	70.09
0.95	86.68	85.22	85.94	74.94	72.47	73.08

4.4 Computational Complexity

To derive an effective computation of this kernel, [4] proposed to use the dynamic programming technique to reduce the complexity of computation to $O(n|S||T|)$. n is the subsequence length and $|S|$ and $|T|$ refer to the number of symbols (words/characters/topics) in S and T. Therefore, the number of symbols in a sequence determines the efficiency of the string kernel. The problem is that the semantic structures of documents cannot be discovered when the number of symbols is greatly reduced. Differently, the topics are a summary of document and can detect the semantic structure of a document.

To further reduce the computational cost, [5] proposed to use the sequence of words instead of characters on the Reuters dataset. The average number of words per document is **141** before removing the stop words and this number drops to **77** after the stop words are removed. In our experiments, the average number of topics is **10**. [5] claimed that if the average sequence length is reduced by about **50%**, the kernel computation time would be reduced by **75%**. By comparison to **77**, the average sequence length is reduced by about **87%** since the topic number of topics is **10** in this paper. Hence, the computational complexity would be greatly reduced.

4.5 Does the Topic Sequential Order Matter?

In this section, we will investigate whether the sequential order of topics matter in text classification. We compared the topic sequence kernel with other kernels including linear, polynomial, RBF kernel and sigmoid kernel. For these kernels, we do not rearrange the topics in a sequential order. We simply use the topic proportions as the weight of each topic. Then each document is represented by a feature vector with topic proportions. Experiments are conducted by varying the number of topics (K) from 5 to 20. From the experimental results, we found that no matter what kind of kernels we used, the performance remains the same for each kernel if the number of topics remains unchanged. Therefore, we name other kernels as *ALL* plus the number of topics. *ALL_5*, for example, indicates the linear, polynomial, RBF or sigmoid kernel with the number of topics being 5. Similarly, *ALL_10*, *ALL_15* and *ALL_20* are either of these kernels with the number of topics being 10, 15 and 20, respectively.

Table 6. Comparison between ***TSK*** and other Kernels

	Micro-average			Macro-average		
	p	r	F	p	r	F
ALL_5	93.03	61.79	74.26	32.66	25.47	28.11
ALL_10	92.38	74.78	**82.65**	74.71	52.27	59.88
ALL_15	95.61	70.25	80.99	71.96	41.31	49.1
ALL_20	96.84	63.69	76.84	61.31	28.8	35.59
TSK	87.06	**86.69**	**86.88**	**76.48**	**75.91**	**75.98**

Table 6 shows that the topic sequence kernel (**TSK**) gives the best micro-average and macro-average F-scores when compared to other kernels. This testifies our hypothesis that the sequential order of topics really matters in text classification using topics. As the Table 6 shows, among the other kernels, *ALL_10* gives the best micro-average and macro-average F-scores when the number of topics is 10.

5 Conclusions and Future Works

Similar to the string kernel and word sequence kernel, topic sequence kernel considers the sequential structure between the symbols (characters/words/topics). In this paper, we first tried to use the topic sequence kernel operating at the topic level to greatly reduce the computational time cost. Initially, the LDA algorithm is used to extract posterior topic distributions in each document and generate the topic sequence based on these topic distributions. Our observations suggest that the optimal result is obtained when the number of topics is equal to the number of categories and the topic stability might be damaged if a document belongs to more than one category.

We focused on topic sequence kernels which are based on the topics. One advantage is that topics are summaries of documents and they can well capture the

semantics of documents. The other advantage is that number of topics per document is much less than the number of words in a document. This can bring the string kernel into practical usage, since string kernel computation is rather resource demanding. Topic sequence kernel is an extension of string kernel. Our work contributes to the structural document representation using topics instead of words or characters and to the reduction of computational runtime cost.

References

1. Joachims, T.: Text Categorization with Support Vector Machines. Technical report, LS VIII NO. 23. University of Dortmund (1997)
2. Osuna, E., Freund, R., Girosi, F.: Training support vector machines: an application to face detection. In: IEEE Conference on Computer Vision and Pattern Recognition, pp. 130–136 (1997)
3. Wang, J.Y.: Application of Support Vector Machines in Bioinformatics. Master's thesis, Dept. Computer Sci. Info. Eng., National Taiwan University (2002)
4. Lodhi, H., Saunders, C., Shawe-Taylor, J., Cristianini, N., Watkins, C.: Text classification using string kernels. The Journal of Machine Learning Research 2, 419–444 (2002)
5. Cancedda, N., Gaussier, E., Goutte, C., Renders, J.M.: Word sequence kernels. Journal of Machine Learning Research 3, 1059–1082 (2003)
6. Shawe-Taylor, J., Cristianini, N.: Kernel Methods for Pattern Analysis. Cambridge University Press (2004)
7. Mercer, J.: Functions of positive and negative type and their connection with the theory of integral equations. Philosophical Transactions of the Royal Society London (A) 209, 415–446 (1909)
8. Deerwester, S., Dumais, S., Landauer, T., Furnas, G., Harshman, R.: Indexing by latent semantic analysis. Journal of the American Society of Information Science 41(6), 391–407 (1990)
9. Hofmann, T.: Probabilistic latent semantic indexing. In: Research and Development in Information Retrieval, pp. 50–57 (1999)
10. Blei, D., Ng, A., Jordan, M.: Latent dirichlet allocation. Journal of Machine Learning Research 3, 993–1022 (2003)
11. Blei, D., Lafferty, J.: Dynamic topic models. In: Proceedings of the 23rd International Conference on Machine Learning, pp. 113–120 (2006)
12. Blei, D., Lafferty, J.: A correlated topic model of science. Annals of Applied Statistics 1(1), 17–35 (2007)
13. Minka, T., Lafferty, J.: Expectation-propagation for the generative aspect model. In: Uncertainty in Artificial Intelligence, UAI (2002)
14. Steyvers, M., Griffiths, T.: Probabilistic topic models. In: Landauer, T., McNamara, D., Dennis, S., Kintsch, W. (eds.) Latent Semantic Analysis: A Road to Meaning. Lawrence Erlbaum (2006)
15. Teh, Y., Newman, D., Welling, M.: A collapsed variational Bayesian inference algorithm for latent Dirichlet allocation. In: Neural Information Processing Systems (2006)