Kernel Latent Semantic Analysis using an Information Retrieval based Kernel

Laurence A. F. Park and Kotagiri Ramamohanarao
Department of Computer Science and Software Engineering,
The University of Melbourne, Australia, 3010
{lapark,kotagiri}@unimelb.edu.au

ABSTRACT

Hidden term relationships can be found within a document collection using Latent semantic analysis (LSA) and can be used to assist in information retrieval. LSA uses the inner product as its similarity function, which unfortunately introduces bias due to document length and term rarity into the term relationships. In this article, we present the novel kernel based LSA method, which uses separate document and query kernel functions to compute document and query similarities, rather than the inner product. We show that by providing an appropriate kernel function, we are able to provide a better fit of our data and hence produce more effective term relationships.

Categories and Subject Descriptors: H.3.3 [Information Search and Retrieval]: Retrieval models

General Terms: Experimentation, Performance

Keywords: Latent Semantic Analysis, Kernel functions

1. INTRODUCTION

Term frequency methods of information retrieval use the occurrence of query terms in a document as the basis of a document similarity score. These methods neglect the fact that there are relationships between terms that could alter the relevance of a document. Understanding a language is a matter of learning term associations, therefore, a method that examines documents for query terms alone is not making any attempt at learning term associations, and therefore is ignoring any benefit that could be obtained by understanding the language.

Latent semantic analysis (LSA) is a linear method of attempting to obtain these term relationships by discovering a small set of hidden *basis terms* that can be used to describe the document collection. A measure of the relationship between each term and basis term is computed, therefore term relationships can be obtained by comparing related basis terms.

LSA has the constraint that the inner product must be used to compute the similarity between documents, and documents and queries. Unfortunately, the inner product is not a desirable document or query similarity metric since it introduces bias from properties such as document lengths and term rarity. The bias leads to longer documents being ranked higher than short documents and common terms having more impact than rare terms. Similarity metrics for document retrieval such as BM25 have been devised in an attempt to remove this bias. Therefore, these metrics should be used for the computation of hidden basis terms.

In this article, we introduce kernel based latent semantic analysis (Kernel LSA). Using Kernel LSA, we are able to choose the functions to use for document-document similarity computations and query-document similarity computations. By choosing appropriate similarity functions, we will assist the Kernel LSA process in its discovery of the hidden basis terms.

Research into latent semantic analysis has been very active in recent years [Efron, 2005, Kontostathis and Pottenger, 2006, Ding, 2005, Farahat and Chen, 2006, Park and Ramamohanarao, 2004]. In each of the cited articles, the methods proposed are forced to weight their term frequency data to remove the bias introduced by LSA's inner product.

Recent methods have used LSA dimension reduction as a kernel for support vector machine classification [Cristianini et al., 2002]. To our knowledge, this paper presents the first study of kernel methods for *unsupervised* information retrieval. This paper contains the following important contributions:

- the formulation of the Kernel LSA process for information retrieval which provides a general framework to incorporate various document and query kernels (Section 3)
- examples of the type of kernel that should be chosen for document and query similarity (Section 3.1)
- a method of approximating kernel LSA to provide acceptable query times on large document sets (Section 3.2).

The article will proceed as follows: Section 2 reviews related work, Section 3 introduces our novel kernel based LSA method, and Section 4 compares Kernel LSA, LSA and BM25 in terms of retrieval precision.

2. RELATED WORK

To our knowledge, there is no previous work examining the effectiveness of using kernel based LSA for unsupervised information retrieval.

LSA is a specific version of Principle component analysis for document vectors. The earliest known application of applying kernels to PCA was shown in the work done on kernel Principle Component Analysis (kernel PCA) [Schölkopf et al., 1998]. PCA has been effectively used in many fields as a method of data smoothing and dimension reduction. The first work on kernel space PCA

[Schölkopf et al., 1998] generalised PCA, allowing us to use any kernel function to compute correlations.

The application of kernels to the field of Information Retrieval document classification was first performed by Christianini *et al.* [Cristianini et al., 2002]. This work on Latent Semantic Kernels describes the process of computing linear kernels for Support Vector Machine (SVM) document classification. Rather than using the inner product as its kernel, the SVM was used with a PCA approximation of the inner product called a latent semantic kernel (LSK). Using this LSK, a smoothed document similarity score could be computed. SVMs were built using the LSK to train and classify documents. There was no analysis of other kernels or application of the SVM with LSK to ad hoc information retrieval.

Our focus is the generalisation of Latent Semantic Analysis to use a kernel function rather than simply the inner product. We examine the choice and application of kernels for document retrieval and not document classification. Therefore, our work is entirely different to, and not an extension of, the work by Christianini *et al.*.

3. USING KERNELS WITH LSA

The SVD in LSA uses an intrinsic inner product to compute the correlation matrix C containing the similarity between document vectors:

$$s(\tilde{d}_m, \tilde{d}_n) = \tilde{d}_m \cdot \tilde{d}_n$$

which is unsuitable for term frequency vectors. The inner product introduces bias into the similarity calculations such as higher scores to longer documents and common terms.

Rather than using the inner product to compute document and query similarity, we should be using the existing similarity metrics that avoid the mentioned bias. In this section, we will show how kernel functions can be applied to LSA such that we are able to define our own similarity functions. We will also show how current information retrieval similarity functions can be used as kernels.

Kernel principal component analysis (Kernel PCA) [Schölkopf et al., 1998] applies the kernel trick, used with support vector machines, to principal component analysis. In this section we will show how we can apply the kernel trick to LSA to perform information retrieval and hence perform Kernel LSA.

Given a query, Kernel LSA document scores are computed using:

$$\tilde{s} = k_q(\tilde{q}, F)U^T I^* U \tag{1}$$

where $\tilde{q}$ is the query vector, F is the term frequency matrix, U is the set of eigenvectors of the gram matrix K, where

$$K = \begin{bmatrix} k_d(\tilde{d}_1, \tilde{d}_1) & k_d(\tilde{d}_1, \tilde{d}_2) & \dots & k_d(\tilde{d}_1, \tilde{d}_D) \\ k_d(\tilde{d}_2, \tilde{d}_1) & k_d(\tilde{d}_2, \tilde{d}_2) & \dots & k_d(\tilde{d}_2, \tilde{d}_D) \\ \vdots & \vdots & \ddots & \vdots \\ k_d(\tilde{d}_D, \tilde{d}_1) & k_d(\tilde{d}_D, \tilde{d}_2) & \dots & k_d(\tilde{d}_D, \tilde{d}_D) \end{bmatrix}$$

and $k_q(\cdot,\cdot)$ and $k_d(\cdot,\cdot)$ are the query-document and document-document similarity functions (kernels) respectively. We will now proceed with the derivation of this equation.

If we have a function $\phi_d(\cdot)$ that is able to remove the bias from our document vectors, and an associated kernel function $k_d(\cdot,\cdot)$, such that $k_d(\tilde{d}_m, \tilde{d}_n) = \phi_d(\tilde{d}_m) \cdot \phi_d(\tilde{d}_n)$ we should apply this function to the document vector set and compute the SVD in order to obtain the set of unbiased basis terms. Therefore, we wish to compute:

$$\phi_d(F) = \Phi = U^T \Sigma V \tag{2}$$

where $K = \Phi\Phi^T$ and $C = \Phi^T\Phi$ are the gram matrix and correlation matrix respectively. The matrices U and Σ^2 are the set of eigenvectors and eigenvalues of the Gram matrix K respectively. Using the decomposition in equation 2, we can obtain an equation for our set of basis terms V:

$$V = \Sigma^{-1} U \Phi$$

Many basis terms with associated low singular values are considered noise. Therefore, we wish to keep the basis terms with high associated singular values and remove the noisy basis terms with low associated singular values. To select wanted basis terms and remove the noisy basis terms, we will pre-multiply V with I^*, an identity matrix with zeros inserted on the diagonal to remove the associated unwanted basis terms:

$$I^* V = I^* \Sigma^{-1} U \Phi$$

We now have an equation for V that cannot be resolved due to the unknown Φ, however, we are able to project vectors into the basis term space. To represent a document vector $\tilde{d}$ in the form of the unbiased basis terms, we must map the vector into the ϕ_d space ($\tilde{d}_\phi = \phi_d(\tilde{d})$) and then into the kernel LSA space (V):

$$\begin{aligned} \tilde{d}_{\phi^*} &= \phi_d(\tilde{d}) V^T I^* \\ &= \phi_d(\tilde{d}) \Phi^T U^T \Sigma^{-1} I^* \\ &= k_d(\tilde{d}, F) U^T \Sigma^{-1} I^* \end{aligned}$$

Where $k_d(\tilde{d}, F)$ is the inner product of the document vector and the document matrix in the document kernel space. If we choose to map the set of document vectors into the basis term space, we use:

$$\begin{aligned} F_{\phi^*} = k_d(F, F) U^T \Sigma^{-1} I^* &= K U^T \Sigma^{-1} I^* \\ &= U^T \Sigma I^* \end{aligned}$$

since $KU^T = U^T\Sigma^2$. To compute the document score, we must map the query into the kernel space and calculate the inner product of the query and documents. The mapped query vector is:

$$\begin{aligned} \tilde{q}_{\phi^*} &= \phi_q(\tilde{q}) V^T \\ &= k_q(\tilde{q}, F) U^T \Sigma^{-1} I^* \end{aligned}$$

where $k_q(\cdot,\cdot)$ is the chosen document-query kernel function. To compute the document scores for query $\tilde{q}$, we use the unbiased document vectors represented using basis terms (F_{ϕ^*}), and the unbiased query vector represented using basis terms ($\tilde{q}_{\phi^*}$), and obtain their inner product:

$$\begin{aligned} \tilde{s} = \tilde{q}_{\phi^*} F_{\phi^*}^T &= k_q(\tilde{q}, F) U^T \Sigma^{-1} I^* \left(U^T \Sigma I^* \right)^T \\ &= k_q(\tilde{q}, F) U^T \Sigma^{-1} I^* \Sigma U \\ &= k_q(\tilde{q}, F) U^T I^* U \end{aligned}$$

giving us the KLSA document scoring function of equation 1.

This implies that if we calculate the eigen decomposition of matrix $K = k_d(F, F)$ to obtain U, we are able to easily compute the score of each document in the kernel feature space.

3.1 Information retrieval kernels

The Kernel LSA query-document score function in equation 1 requires us to choose a document-document similarity kernel ($k_d(\cdot,\cdot)$, in order to calculate K and then U), and a query-document similarity kernel ($k_q(\cdot,\cdot)$).

A sparse kernel produces zero if the input vectors have no features in common. By choosing a sparse kernel for k_q, we only have to scan through a portion of the document mapping $U^T I^* U$ to compute the final document scores, and hence obtain a faster query time. By choosing a sparse kernel for k_d, the Gram matrix

K will contain many zeros, and hence its eigenvalue decomposition is faster to compute. Kernels such as a radial basis kernel and a polynomial kernel with non-zero offset are not sparse.

In this section, we will examine the BM25 document and query similarity functions, which can be used as sparse kernel functions.

3.1.1 *BM25 probabilistic similarity function*

The current state of the art in query-document similarity is BM25 [Jones et al., 2000]:

$$k_q(\tilde{q},\tilde{d}) = \sum_{t\in Q}\left(\frac{(k_1+1)f_{d,t}}{K+f_{d,t}}\right)w_t \qquad (3)$$

where document-document similarity is calculated using:

$$k_d(\tilde{d}_m,\tilde{d}_n) = \sum_{t\in(d_m\cap d_n)}\left(\frac{(k_1+1)f_{d_m,t}}{K_m+f_{d_m,t}}\right)\left(\frac{(k_1+1)f_{d_n,t}}{K_n+f_{d_n,t}}\right)w_t \qquad (4)$$

where k_1 is a constant (usually set to 2), K_m and K_n are used to normalise the document length and are computed using $K_i = k_1\left((1-b)+b\frac{dl_i}{dl_{\text{avg}}}\right)$ where b is a constant (usually 0.75), dl_i is the length of document i and dl_{avg} is the average document length, $w_t = \log\left(\frac{N-f_t+0.5}{f_t+0.5}\right)$, N is the number of documents and f_t is the number of documents containing term t.

3.1.2 *Choice of kernel functions*

For each of these similarity calculation methods and for others that have not been mentioned here, we are able to state that there exists a document vector normalisation function $\phi_d(\cdot)$ and a query normalisation function $\phi_q(\cdot)$ that are unknown, but when used in an inner product with each other, produce the wanted document-document and query-document similarity functions. Therefore:

$$k_q(\tilde{q},\tilde{d}) = \phi_q(\tilde{q})\cdot\phi_d(\tilde{d}), \quad k_d(\tilde{d}_m,\tilde{d}_n) = \phi_d(\tilde{d}_m)\cdot\phi_d(\tilde{d}_n)$$

implying that the similarity functions can be used as kernels.

Due to the nature of kernel functions and their relationship to their associated mapping function, there is little restriction on the choice of our kernel. For example, we can choose some mapping functions $\phi_q(\cdot)$ and $\phi_d(\cdot)$ such that $k_d(\cdot,\cdot)$ is the TF-IDF document-document similarity function and $k_q(\cdot,\cdot)$ is the BM25 query-document similarity function. Therefore, we do not have to use related $k_d(\cdot,\cdot)$ and $k_q(\cdot,\cdot)$. This allows us to choose the best similarity functions for each task. Other forms of document-document similarity functions can be found in [Lee et al., 2005, Lewis et al., 2006, Park and Ramaohanarao, 2008]. Note that the kernels suggested are significantly different to kernels used previously in Kernel PCA.

3.2 Kernel LSA query time analysis

Kernel LSA allows us to combine the documents and basis terms into a document expansion:

$$\tilde{s} \quad = k_q(\tilde{q},F)D$$

where $D = U^T I^* U$ is a document expansion. Given an arbitrary query kernel k_q, we are unable to perform any more computation until a query is supplied. Therefore the query process consists of computing the query-document kernel value for each document $(k_q(\tilde{q},F))$ and then applying these values to the document expansion D. This is clearly inefficient since we have to process every document and then perform the expansion on every document before we can obtain a document score, making our query time dependent on the number of documents in the index.

To provide acceptable query times, we propose the use of a vector thresholding function $\text{top}_z()$ which sets all but the greatest z vector elements to zero. When using this thresholding function, we can compute approximate kernel LSA document scores with:

$$\tilde{s} \quad = \text{top}_z(k_q(\tilde{q},F))\,D$$

When using this function, we only need to compute the top z values from $k_q(\tilde{q},F)$ and use z columns of D.

In Section 4, we will examine the effect of varying z. By using an information retrieval based kernel, we are able to efficiently approximate the top document kernel scores and drastically reduce the processing time required during query time.

3.3 Smoothing the Kernel LSA model

Analysis has shown that the effectiveness of LSA reduces as the size of the document collection grows [Park and Ramamohanarao, 2009a]. To avoid this problem, smoothing is applied to the model [Park and Ramamohanarao, 2009b, 2007].

To smooth the kernel LSA model, we mix the model scores with the scores computed using the query-document similarity function:

$$\begin{aligned}
&\alpha k_q(\tilde{q},F)U^T I^* U + (1-\alpha)k_q(\tilde{q},F)\\
&\qquad = k_q(\tilde{q},F)\left[\alpha U^T I^* U + (1-\alpha)I\right]\\
&\qquad = k_q(\tilde{q},F)\left[\alpha U^T I^* U + (1-\alpha)U^T I U\right]\\
&\qquad = k_q(\tilde{q},F)U^T\left[\alpha I^* + (1-\alpha)I\right]U
\end{aligned}$$

where $\alpha \in [0,1]$ is the mixing parameter. We can see that the smoothing is simply modifying the contribution of each of the computed eigenvectors, or basis terms. Using kernel LSA, we have selected x basis terms by applying I^*, which sets all of the remaining basis terms to zero. After applying the smoothing, it is as though we have introduced the removed basis terms with the weight $1-\alpha$. Also note that if $\alpha = 0$ our scoring function is simply $k_q(\tilde{q},F)$.

4. RETRIEVAL ACCURACY

In this section we will examine the effect of kernel LSA on retrieval accuracy. Our variables are the choice of kernel, the smoothing parameter α (from Section 3.3), and the thresholding value z (from Section 3.2). We will compare our kernel LSA methods to latent semantic indexing (LSI) and the state of the art probabilistic measure BM25. Each of our experiments using kernel LSA and LSI use the eigenvectors associated to the greatest 300 eigenvalues, as suggested in [Deerwester et al., 1990].

Each or our retrieval results will be measured using mean average precision. Mean average precision (MAP) is the mean of the set of AP scores from each query. MAP is an information retrieval evaluation metric that can also be thought of as the area under the precision-recall curve of a system. A MAP score of 1 implies that all of the relevant documents were ranked higher than the irrelevant documents across all queries.

Our experiments were performed on the SMART document collection, containing the document sets CRAN, CISI, CACM, and MED[1]. These document sets have been used extensively to show the performance of our baseline, LSI [Efron, 2005, Kontostathis and Pottenger, 2006, Ding, 2005, Farahat and Chen, 2006, Park and Ramamohanarao, 2004]. The number of documents in each document set is 1398, 1460, 3204 and 1033 respectively.

We will examine the effect on MAP when using our information retrieval kernels (BM25 document-document kernel and BM25

[1]Available from ftp://ftp.cs.cornell.edu/pub/smart

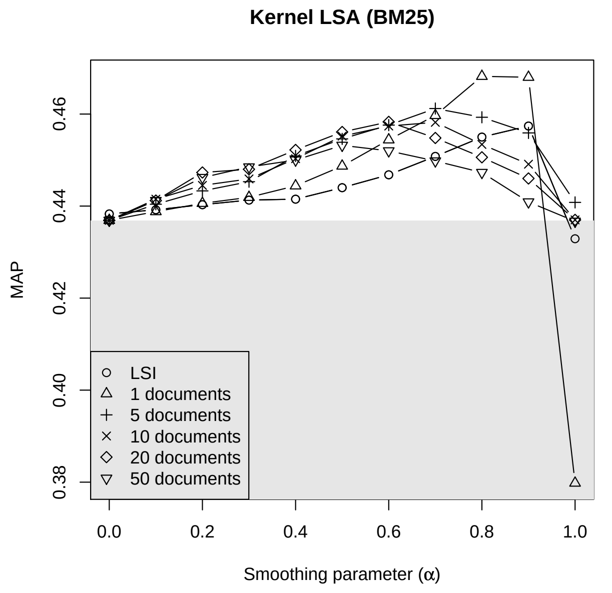

Figure 1: Mean average precision (MAP) using Kernel LSA with a BM25 document-document kernel. The baselines shown are BM25 (shown by the gray region) and LSI. Each set of points shows the MAP produced when thresholding to z documents.

query-document kernel shown in equations 4 and 3 respectively) as k_d and k_q. We have provided a plot in Figure 1 of the mean average precision obtained when varying the smoothing parameter α, for $z = 1$, 5, 10, 20 and 50. The results shown are from the CRAN document set. The results from the CISI, CACM and MED document sets all exhibited similar trends.

From this plot, we can see that we achieve the greatest mean average precision when using $z = 1$ with a smoothing parameter of $\alpha = 0.9$. We can see that when the smoothing parameter is set to 0.9, the MAP increases as z decreases.

We also found that our more efficient information retrieval based kernels provides us with an increase in MAP over LSI and BM25 when using only an approximation of the top document scores. This implies that we are able to use this approximation to obtain fast query times, while still benefiting from the kernel LSA relationships.

5. CONCLUSION

LSA uses the inner product as its similarity metric; this unfortunately introduces bias into the basis term computation due to poor normalisation of document lengths and rare terms.

In this article, we have formalised the kernel-based LSA method for computing basis terms to be used for unsupervised document retrieval. By using kernel LSA, we are able to choose appropriate document-document and query-document similarity functions to allow the kernel LSA process to discover more effective basis terms. Unfortunately, kernel based LSA produces infeasible query times, therefore we also presented an approximation to kernel based LSA that provides acceptable query times.

We examined the retrieval performance of a BM25 kernel and found that the information retrieval based kernel, used with our approximate kernel based LSA, increased the retrieval precision. Using the BM25 kernel functions, we are able to compute a set of non-linear basis terms that provide a closer fit to the document set that is unachievable using the linear basis terms found with LSA.

References

Nello Cristianini, John Shawe-Taylor, and Huma Lodhi. Latent semantic kernels. *J. Intell. Inf. Syst.*, 18(2-3):127–152, 2002. ISSN 0925-9902.

S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. A. Harshman. Indexing by latent semantic analysis. *Journal of the the American Society for Information Science*, 41:391–407, 1990.

Chris H.Q. Ding. A probabilistic model for latent semantic indexing. *Journal of the American Society for Information Science and Technology*, 56(6):597–608, 2005.

Miles Efron. Eigenvalue-based model selection during latent semantic indexing. *JASIST*, 56(9):969–988, 2005.

A. O. Farahat and F. R. Chen. Improving probabilistic latent semantic analysis using principal component analysis. In *Eleventh Conference of the European Chapter of the Association for Computational Linguistics (EACL -2006)*, 2006.

K. Sparck Jones, S. Walker, and S. E. Robertson. A probabilistic model of information retrieval: development and comparative experiments, part 2. *Information Processing and Management*, 36(6):809—840, 2000. ISSN 0306-4573. doi: 10.1016/S0306-4573(00)00015-7.

April Kontostathis and William M. Pottenger. A framework for understanding latent semantic indexing (LSI) performance. *Inf. Process. Manage.*, 42(1):56–73, 2006. ISSN 0306-4573. doi: 10.1016/j.ipm.2004.11.007.

Michael D. Lee, Brandon Pincombe, and Matthew Welsh. An empirical evaluation of models of text document similarity. In *CogSci2005*, pages 1254–1259, 2005.

James Lewis, Stephan Ossowski, Justin Hicks, Mounir Errami, and Harold R. Garner. Text similarity: an alternative way to search MEDLINE. *Bioinformatics*, 22(18):2298–2304, 2006. doi: 10.1093/bioinformatics/btl388.

Laurence A. F. Park and Kotagiri Ramamohanarao. Query expansion using a collection dependent probabilistic latent semantic thesaurus. In Zhi-Hua Zhou, Hang Li, and Qiang Yang, editors, *The Eleventh Pacific-Asia Conference on Knowledge Discovery and Data Mining Workshop*, volume 4426 of *Lecture Notes in Computer Science*, pages 224–235. Springer, 2007. ISBN 978-3-540-71700-3. doi: 10.1007/978-3-540-71701-0_24.

Laurence A. F. Park and Kotagiri Ramamohanarao. Hybrid pre-query term expansion using latent semantic analysis. In Rajeev Rastogi, Katharina Morik, Max Bramer, and Xindong Wu, editors, *The Fourth IEEE International Conference on Data Mining*, pages 178–185, Los Alamitos, California, November 2004. IEEE Computer Society. doi: 10.1109/ICDM.2004.10085.

Laurence A. F. Park and Kotagiri Ramamohanarao. An analysis of latent semantic term self-correlation. *ACM Transactions on Information Systems*, 27(2):1–35, 2009a. ISSN 1046-8188. doi: 10.1145/1462198.1462200.

Laurence A. F. Park and Kotagiri Ramamohanarao. Efficient storage and retrieval of probabilistic latent semantic information for information retrieval. *The International Journal on Very Large Data Bases*, 18(1): 141–156, January 2009b. doi: 10.1007/s00778-008-0093-2.

Laurence A. F. Park and Kotagiri Ramaohanarao. The effect of weighted term frequencies on probabilistic latent semantic term relationships. In Amihood Amir, Andrew Turpin, and Alistair Moffat, editors, *The 5th String Processing and Information Retrieval Symposium*, volume 5280/2009, pages 63–74, 2008. doi: 10.1007/978-3-540-89097-3_8.

B. Schölkopf, A. Smola, and K.-R. Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10:1299–1319, 1998.