Regularized Latent Semantic Indexing

Quan Wang
MOE-Microsoft Key Laboratory of Statistics&Information Technology
Peking University, China
v-quwan@microsoft.com

Jun Xu, Hang Li
Microsoft Research Asia
No. 5 Danling Street
Beijing, China
{junxu,hangli}@microsoft.com

Nick Craswell
Microsoft
Bellevue, Washington, USA
nickcr@microsoft.com

ABSTRACT

Topic modeling can boost the performance of information retrieval, but its real-world application is limited due to scalability issues. Scaling to larger document collections via parallelization is an active area of research, but most solutions require drastic steps such as vastly reducing input vocabulary. We introduce Regularized Latent Semantic Indexing (RLSI), a new method which is designed for parallelization. It is as effective as existing topic models, and scales to larger datasets without reducing input vocabulary. RLSI formalizes topic modeling as a problem of minimizing a quadratic loss function regularized by ℓ_1 and/or ℓ_2 norm. This formulation allows the learning process to be decomposed into multiple sub-optimization problems which can be optimized in parallel, for example via MapReduce. We particularly propose adopting ℓ_1 norm on topics and ℓ_2 norm on document representations, to create a model with compact and readable topics and useful for retrieval. Relevance ranking experiments on three TREC datasets show that RLSI performs better than LSI, PLSI, and LDA, and the improvements are sometimes statistically significant. Experiments on a web dataset, containing about 1.6 million documents and 7 million terms, demonstrate a similar boost in performance on a larger corpus and vocabulary than in previous studies.

Categories and Subject Descriptors

H.3 [**Information Storage and Retrieval**]: Content Analysis and Indexing

General Terms

Experimentation

Keywords

Topic Modeling, Regularization, Sparse Methods

1. INTRODUCTION

Recent years have seen significant progress on topic modeling technologies in machine learning, information retrieval, natural language processing, and other related fields. Given a collection of text documents, a topic model represents the relationship between terms and documents through latent topics. A topic is defined as a probability distribution of terms or a cluster of weighted terms. A document is then viewed as a bag of terms generated from a mixture of latent topics. Various topic modeling methods, such as Latent Semantic Indexing (LSI) [10], Probabilistic Latent Semantic Indexing (PLSI) [16], and Latent Dirichlet Allocation (LDA) [3] have been proposed and successfully applied in various settings.

One of the main challenges in topic modeling is scaling to millions or even billions of *documents* while maintaining a representative vocabulary of *terms*, which is necessary in many applications such as web search. A typical approach is to approximate the learning processes of an existing topic model.

In this work, instead of modifying existing methods, we introduce a new topic modeling method that is intrinsically scalable: Regularized Latent Semantic Indexing (RLSI). Topic modeling is formalized as minimization of a quadratic loss function on term-document occurrences regularized by ℓ_1 and/or ℓ_2 norm. Specifically, in RLSI the text collection is represented as a term-document matrix, where each entry represents the occurrence (or tf-idf score) of a term in a document. The term-document matrix is then approximated by the product of two matrices: the term-topic matrix which represents the latent topics with terms and the topic-document matrix which represents the documents with topics. Finally, the quadratic loss function is defined as the squared Frobenius norm of the difference between the term-document matrix and the output of the topic model. Both ℓ_1 norm and ℓ_2 norm may be used for regularization. We particularly propose using ℓ_1 norm on topics and ℓ_2 norm on document representations, which can result in a model with compact and readable topics and useful for retrieval. Note that we call our new method RLSI because it makes use of the same quadratic loss function as LSI. RLSI differs from LSI in that it uses regularization rather than orthogonality to constrain the solutions.

The learning process of RLSI iteratively updates the term-topic matrix given the fixed topic-document matrix, and updates the topic-document matrix given the fixed term-topic matrix. The formulation of RLSI makes it possible to decompose the learning problem into multiple sub-optimization problems and conduct learning in parallel. Specifically, for both the term-topic matrix and the topic-document matrix, the update in each iteration is decomposed into many sub-optimization problems. These may be run in parallel, which is the main reason that RLSI can scale up. We describe our implementation of RLSI in MapReduce [9]. The MapReduce system maps the sub-optimization problems over multiple processors and then merges (reduces) the results from the processors. During this process, documents and terms are distributed and processed automatically.

For probabilistic topic models like LDA and PLSI, the scalability challenge mainly comes from the necessity of simultaneously updating the term-topic matrix to meet the probability distribution assumptions. When the number of terms is large, which is inevitable in real applications, this problem becomes particularly severe. For LSI, the challenge is due to the orthogonality assumption in the formulation, and as a result the problem needs to be solved by Singular Value Decomposition (SVD) and thus is hard to be parallelized.

Regularization is a well-known technique in machine learning. In our setting, if we employed ℓ_2 norm on topics and ℓ_1 norm on document representations, RLSI becomes Sparse Coding [19, 25], which is a method used in computer vision and other fields. As far as we know, regularization for topic modeling has not been widely studied, in terms of the performance of different norms or their scalability advantages.

Experimental results on a large web dataset show that 1) RLSI can scale up well and help improve search relevance. Specifically, we show that RLSI can efficiently run on *1.6 million documents and 7 million terms* on 16 distributed machines. In contrast, existing methods on parallelizing LDA were demonstrated on far fewer documents and/or far fewer terms. Experiments on three TREC datasets show that 2) The readability and coherence of RLSI topics is equal or better than those learned by LDA, PLSI and LSI. 3) RLSI topics can be used in retrieval with better performance than LDA, PLSI, and LSI (sometimes statistically significant). 4) The best choice of regularization is ℓ_1 on topics and ℓ_2 on document representations in terms of topic readability and retrieval performance.

2. RELATED WORK

Studies on topic modeling fall into two categories: probabilistic approaches and non-probabilistic (matrix factorization) approaches. In the probabilistic approaches, a topic is defined as a probability distribution over terms and documents are defined as data generated from mixtures of topics. To generate a document, one chooses a distribution over topics. Then, for each term in that document, one chooses a topic according to the topic distribution, and draws a term from the topic according to its term distribution. For example, PLSI [16] and LDA [3] are two widely-used generative models. In the non-probabilistic approaches, the term-document matrix is projected into a K-dimensional topic space in which each axis corresponds to a topic. In the topic space, each document is represented as a linear combination of the K topics. LSI [10] is a representative non-probabilistic model. It decomposes the term-document matrix with SVD under the assumption that topics are orthogonal. See also Non-negative Matrix Factorization (NMF) [17, 18] and Sparse Coding methods [19, 25].

It has been demonstrated that topic modeling is useful for knowledge discovery, relevance ranking in search, and document classification [23, 35]. In fact, topic modeling is becoming one of important technologies in machine learning, information retrieval, and other related fields.

Most efforts to improve topic modeling scalability have modified existing learning methods, such as LDA. Newman, et al. [24] proposed Approximate Distributed LDA (AD-LDA), in which each processor performs a local Gibbs sampling iteration followed by a global update. Two recent papers implemented AD-LDA as PLDA [34] and modified AD-LDA as PLDA+ [21], using MPI [32] and MapReduce [9]. In [2], the authors proposed purely asynchronous distributed LDA algorithms based on Gibbs Sampling or Bayesian inference, called Async-CGB or Async-CVB, respectively. In Async-CGB and Async-CVB, each processor performs a local computation step followed by a step of communicating with other processors. In all the methods, the local processors need to maintain and update a dense term-topic matrix, usually in memory, which becomes a bottleneck for improving the scalability. In this paper, we propose a new topic model learning algorithm which can efficiently scale up to large text corpora. The key ingredient of our method is to make the formulation of learning decomposable and thus make the process of learning parallelizable. In [1, 15], online versions of stochastic LDA were proposed. In this paper, we consider batch learning of topic models, which is a different setting from online learning. For other related work refer to [23, 31, 36].

Regularization is a common technique in machine learning to prevent over-fitting. Typical examples of regularization in machine learning include the use of ℓ_1 and ℓ_2 norms. Regularization via ℓ_1 norm uses the sum of absolute values of parameters and thus has the effect of causing many parameters to be zero and selecting a sparse model as solution [14, 26]. Regularization via ℓ_2 norm, on the other hand, uses the sum of squares of parameters and thus can make a smooth regularization and effectively deal with over-fitting.

Sparse methods have recently received a lot of attention in machine learning community. They aim to learn sparse representations (simple models) hidden in the input data by using ℓ_1 norm regularization. Sparse Coding algorithms [19, 25] are proposed which can be used for discovering basis functions, to capture meta-level features in the input data. One justification to the sparse methods is that human brains have similar sparse mechanism for information processing. For example, when Sparse Coding algorithms are applied to natural images, the learned bases resemble the receptive fields of neurons in the visual cortex [25]. Previous work on sparse methods mainly focused on image processing (e.g., [28]). In this paper we propose using sparse methods (ℓ_1 norm regularization) in topic modeling, particularly to make the learned topics sparse. The use of sparse methods for topic modeling was also proposed very recently by Chen et al. [8]. Their motivation was not to improve scalability and they made an orthogonality assumption (requiring an SVD). In [33], the authors also proposed to discover sparse topics based on a modified version of LDA.

3. SCALABILITY OF TOPIC MODELS

One of the key problems in topic modeling is to improve scalability, to handle millions of documents or even more. As collection size increases, so does vocabulary size, rather than a maximum vocabulary being reached. For example, in the 1.6 million web documents in our experiment, there are more than 7 million unique terms even after pruning the low frequency ones (e.g., with term frequency in the whole collection less than 2). This means that both matrices, term-topic and topic-document, grow as the number of documents increases.

LSI needs to be solved by SVD due to the orthogonality assumption. The time complexity of computing SVD is normally of order $O(\min\{MN^2, NM^2\})$, where M denotes number of rows of the input matrix and N number of columns. Thus, it appears to be very difficult to make LSI scalable and efficient.

For PLSI and LDA, it is necessary to maintain the probability distribution constraints of the term-topic matrix. When the matrix is large, there is a cost for maintaining the probabilistic framework. One possible solution is to reduce the number of terms, but the negative consequence is that it can sacrifice learning accuracy.

How to make existing topic modeling methods scalable is still a challenging problem. In this paper, we adopt a different approach, that is, to develop new methods which can work equally well or even better, but are scalable by design.

4. RLSI

Table 1: Table of notations.

Notation	Meaning
M	Number of terms in vocabulary
N	Number of documents in collection
K	Number of topics
$\mathbf{D} \in \mathbb{R}^{M \times N}$	Term-document matrix $[\boldsymbol{d}_1, \cdots, \boldsymbol{d}_N]$
$\boldsymbol{d}_n$	The n^{th} document
d_{mn}	Weight of the m^{th} term in document $\boldsymbol{d}_n$
$\mathbf{U} \in \mathbb{R}^{M \times K}$	Term-topic matrix $[\boldsymbol{u}_1, \cdots, \boldsymbol{u}_K]$
$\boldsymbol{u}_k$	The k^{th} topic
u_{mk}	Weight of the m^{th} term in topic $\boldsymbol{u}_k$
$\mathbf{V} \in \mathbb{R}^{K \times N}$	Topic-document matrix $[\boldsymbol{v}_1, \cdots, \boldsymbol{v}_N]$
$\boldsymbol{v}_n$	Representation of $\boldsymbol{d}_n$ in the topic space
v_{kn}	Weight of the k^{th} topic in $\boldsymbol{v}_n$

4.1 Problem Formulation

We are given a set of documents $\mathcal{D}$ with size N, containing terms from a vocabulary $\mathcal{V}$ with size M. A document is simply represented as an M-dimensional vector $\boldsymbol{d}$, where the m^{th} entry denotes the score of the m^{th} term, for example, a Boolean value indicating occurrence, term frequency, tf-idf, or joint probability of the term and document. The N documents in $\mathcal{D}$ are then represented in an $M \times N$ term-document matrix $\mathbf{D} = [\boldsymbol{d}_1, \cdots, \boldsymbol{d}_N]$, in which each row corresponds to a term and each column corresponds to a document.

A topic is defined over terms in the vocabulary and is also represented as an M-dimensional vector $\boldsymbol{u}$, where the m^{th} entry denotes the weight of the m^{th} term in the topic. Intuitively, the terms with larger weights are more indicative to the topic. Suppose that there are K topics in the collection. The K topics can be summarized into an $M \times K$ term-topic matrix $\mathbf{U} = [\boldsymbol{u}_1, \cdots, \boldsymbol{u}_K]$, in which each column corresponds to a topic.

Topic modeling means discovering the latent topics in the document collection as well as modeling the documents by representing them as mixtures of the topics. More precisely, given topics $\boldsymbol{u}_1, \cdots, \boldsymbol{u}_K$, document $\boldsymbol{d}_n$ is succinctly represented as $\boldsymbol{d}_n \approx \sum_{k=1}^{K} v_{kn}\boldsymbol{u}_k = \mathbf{U}\boldsymbol{v}_n$, where v_{kn} denotes the weight of the k^{th} topic $\boldsymbol{u}_k$ in document $\boldsymbol{d}_n$. The larger value of v_{kn}, the more important role topic $\boldsymbol{u}_k$ plays in the document. Let $\mathbf{V} = [\boldsymbol{v}_1, \cdots, \boldsymbol{v}_N]$ be the topic-document matrix, where column $\boldsymbol{v}_n$ stands for the representation of document $\boldsymbol{d}_n$ in the latent topic space. Table 1 gives a summary of notations.

Different topic modeling techniques choose different schemas to model matrices $\mathbf{U}$ and $\mathbf{V}$ and impose different constraints on them. For example, in the generative topic models such as PLSI and LDA, $\boldsymbol{u}_1, \cdots, \boldsymbol{u}_K$ are probability distributions so that $\sum_{m=1}^{M} u_{mk} = 1$ for $k = 1, \cdots, K$; In LSI, topics $\boldsymbol{u}_1, \cdots, \boldsymbol{u}_K$ are orthogonal and thus SVD can be applied.

Regularized Latent Semantic Indexing (RLSI) learns latent topics as well as representations of documents from the given text collections in the following way.

Document $\boldsymbol{d}_n$ is approximated as $\mathbf{U}\boldsymbol{v}_n$ where $\mathbf{U}$ is the term-topic matrix and $\boldsymbol{v}_n$ is the representation of $\boldsymbol{d}_n$ in the latent topic space. The goodness of the approximation is measured by the squared ℓ_2 norm of the difference between $\boldsymbol{d}_n$ and $\mathbf{U}\boldsymbol{v}_n$: $\|\boldsymbol{d}_n - \mathbf{U}\boldsymbol{v}_n\|_2^2$. Furthermore, regularization is made on topics and document representations. Specifically, we suggest ℓ_1 norm regularization on term-topic matrix $\mathbf{U}$ (i.e., topics $\boldsymbol{u}_1, \cdots, \boldsymbol{u}_K$) and ℓ_2 on topic-document matrix $\mathbf{V}$ (i.e., document representations $\boldsymbol{v}_1, \cdots, \boldsymbol{v}_N$) to favor a model with compact and readable topics and useful for retrieval.

Thus, given a text collection $\mathcal{D} = \{\boldsymbol{d}_1, \ldots, \boldsymbol{d}_N\}$, RLSI amounts to solving the following optimization problem:

$$\min_{\mathbf{U},\{\boldsymbol{v}_n\}} \sum_{n=1}^{N} \|\boldsymbol{d}_n - \mathbf{U}\boldsymbol{v}_n\|_2^2 + \lambda_1 \sum_{k=1}^{K} \|\boldsymbol{u}_k\|_1 + \lambda_2 \sum_{n=1}^{N} \|\boldsymbol{v}_n\|_2^2, \qquad (1)$$

Algorithm 1 Regularized Latent Semantic Indexing

```
Require: $\mathbf{D} \in \mathbb{R}^{M \times N}$
1: $\mathbf{V}^{(0)} \in \mathbb{R}^{K \times N} \leftarrow$ random matrix
2: for $t = 1 : T$ do
3:     $\mathbf{U}^{(t)} \leftarrow$ Update$\mathbf{U}(\mathbf{D}, \mathbf{V}^{(t-1)})$
4:     $\mathbf{V}^{(t)} \leftarrow$ Update$\mathbf{V}(\mathbf{D}, \mathbf{U}^{(t)})$
5: end for
6: return $\mathbf{U}^{(T)}, \mathbf{V}^{(T)}$
```

where $\lambda_1 \geq 0$ is the parameter controlling the regularization on $\boldsymbol{u}_k$: the larger value of λ_1, the more sparse $\boldsymbol{u}_k$; and $\lambda_2 \geq 0$ is the parameter controlling the regularization on $\boldsymbol{v}_n$: the larger value of λ_2, the larger amount of shrinkage on $\boldsymbol{v}_n$.

In general, the regularization on topics and document representations (the second term and the third term) can be either ℓ_1 norm or ℓ_2 norm. When they are ℓ_2 and ℓ_1 respectively, the method is equivalent to Sparse Coding [19, 25]. When both of them are ℓ_1, the model is similar to the double sparse model proposed in [28][1].

4.2 Regularization Strategy

We propose using the formulation above (i.e., regularization via ℓ_1 norm on topics and ℓ_2 norm on document representations), because in our experience this regularization strategy leads to a model with compact and readable topics and useful for retrieval.

First, ℓ_1 norm regularization on topics has the effect of making them compact. We do this under the assumption that the essence of a topic can be captured via a small number of terms, which is reasonable in practice. In many applications, small and concise topics are more useful. For example, small topics can be interpreted as sets of synonyms, roughly corresponding to the WordNet synsets used in natural language processing.

Second, ℓ_1 norm can make the topics readable, no matter whether it is imposed on topics or document representations, according to our experiments. This has advantages in applications such as text summarization and visualization.

Third, there are four ways of combining ℓ_1 and ℓ_2 norms. We perform retrieval experiments across multiple test collections, showing that better ranking performance is achieved with ℓ_1 norm on topics and ℓ_2 norm on document representations.

Last, in both learning and using of topic models, topic sparsity means that we can efficiently store and process topics. We can also leverage existing techniques on sparse matrix computation [4, 20], which are efficient and scalable.

4.3 Optimization

The optimization Eq. (1) is convex with respect to $\mathbf{U}$ when $\mathbf{V}$ is fixed and convex with respect to $\mathbf{V}$ when $\mathbf{U}$ is fixed. However, it is not convex with respect to both of them. Following the practice in Sparse Coding [19], we optimize the function in Eq. (1) by alternately minimizing it with respect to term-topic matrix $\mathbf{U}$ and topic-document matrix $\mathbf{V}$. This procedure is summarized in Algorithm 1. Note that for simplicity we describe the algorithm when ℓ_1 norm is imposed on topics and ℓ_2 norm on document representations; one can easily extend it to other regularization strategies.

4.3.1 *Update of Matrix* $\mathbf{U}$

Holding $\mathbf{V} = [\boldsymbol{v}_1, \cdots, \boldsymbol{v}_N]$ fixed, the update of $\mathbf{U}$ amounts to the

[1]Note that both Sparse Coding and double sparse model formulate optimization problems in constrained forms instead of regularized forms. The two forms are equivalent.

following optimization problem:

$$\min_{\mathbf{U}} \|\mathbf{D} - \mathbf{U}\mathbf{V}\|_F^2 + \lambda_1 \sum_{m=1}^{M} \sum_{k=1}^{K} |u_{mk}|, \quad (2)$$

where $\|\cdot\|_F$ is the Frobenius norm and u_{mk} is the $(mk)^{th}$ entry of $\mathbf{U}$. Let $\bar{\boldsymbol{d}}_m = (d_{m1}, \cdots, d_{mN})^T$ and $\bar{\boldsymbol{u}}_m = (u_{m1}, \cdots, u_{mK})^T$ be the column vectors whose entries are those of the m^{th} row of $\mathbf{D}$ and $\mathbf{U}$ respectively. Thus, Eq. (2) can be rewritten as

$$\min_{\{\bar{\boldsymbol{u}}_m\}} \sum_{m=1}^{M} \left\|\bar{\boldsymbol{d}}_m - \mathbf{V}^T \bar{\boldsymbol{u}}_m\right\|_2^2 + \lambda_1 \sum_{m=1}^{M} \|\bar{\boldsymbol{u}}_m\|_1,$$

which can be decomposed into M optimization problems that can be solved independently, with each corresponding to one row of $\mathbf{U}$:

$$\min_{\bar{\boldsymbol{u}}_m} \left\|\bar{\boldsymbol{d}}_m - \mathbf{V}^T \bar{\boldsymbol{u}}_m\right\|_2^2 + \lambda_1 \|\bar{\boldsymbol{u}}_m\|_1, \quad (3)$$

for $m = 1, \cdots, M$.

Eq. (3) is an ℓ_1-regularized least squares problem, whose objective function is not differentiable and it is not possible to directly apply gradient-based methods. A number of techniques can be used here, such as interior point method [7], coordinate descent with soft-thresholding [13, 14], Lars-Lasso algorithm [12, 26], and feature-sign search [19]. Here we choose coordinate descent with soft-thresholding.

Let $\bar{\boldsymbol{v}}_k = (v_{k1}, \cdots, v_{kN})^T$ be the column vector whose entries are those of the k^{th} row of $\mathbf{V}$, $\mathbf{V}^T{}_{\setminus k}$ the matrix of $\mathbf{V}^T$ with the k^{th} column removed, and $\bar{\boldsymbol{u}}_{m\setminus k}$ the vector of $\bar{\boldsymbol{u}}_m$ with the k^{th} entry removed, and we can rewrite the objective function in Eq.(3) as

$$\begin{aligned} L(\bar{\boldsymbol{u}}_m) &= \left\|\bar{\boldsymbol{d}}_m - \mathbf{V}^T{}_{\setminus k}\bar{\boldsymbol{u}}_{m\setminus k} - u_{mk}\bar{\boldsymbol{v}}_k\right\|_2^2 + \lambda_1 \left\|\bar{\boldsymbol{u}}_{m\setminus k}\right\|_1 + \lambda_1 |u_{mk}| \\ &= u_{mk}^2 \|\bar{\boldsymbol{v}}_k\|_2^2 - 2u_{mk}\left(\bar{\boldsymbol{d}}_m - \mathbf{V}^T{}_{\setminus k}\bar{\boldsymbol{u}}_{m\setminus k}\right)^T \bar{\boldsymbol{v}}_k + \lambda_1 |u_{mk}| + const \\ &= u_{mk}^2 s_{kk} - 2u_{mk}\left(r_{mk} - \sum_{l\neq k} s_{kl}u_{ml}\right) + \lambda_1 |u_{mk}| + const, \end{aligned}$$

where s_{ij} and r_{ij} are the $(ij)^{th}$ entries of $K \times K$ matrix $\mathbf{S} = \mathbf{V}\mathbf{V}^T$ and $M \times K$ matrix $\mathbf{R} = \mathbf{D}\mathbf{V}^T$, respectively, and $const$ is a constant with respect to u_{mk}. Then, we can conduct the minimization over u_{mk} while keeping all the u_{ml} fixed for which $l \neq k$. Furthermore, $L(\bar{\boldsymbol{u}}_m)$ is differentiable with respect to u_{mk} except for the point $u_{mk} = 0$. Forcing the partial derivative to be zero leads to

$$u_{mk} = \begin{cases} \dfrac{\left(r_{mk} - \sum_{l\neq k} s_{kl}u_{ml}\right) - \frac{1}{2}\lambda_1}{s_{kk}}, & \text{if } u_{mk} > 0, \\ \dfrac{\left(r_{mk} - \sum_{l\neq k} s_{kl}u_{ml}\right) + \frac{1}{2}\lambda_1}{s_{kk}}, & \text{if } u_{mk} < 0, \end{cases}$$

which can be approximated by the following update rule:

$$u_{mk} \leftarrow \frac{\left(\left|r_{mk} - \sum_{l\neq k} s_{kl}u_{ml}\right| - \frac{1}{2}\lambda_1\right)_+ \operatorname{sign}\left(r_{mk} - \sum_{l\neq k} s_{kl}u_{ml}\right)}{s_{kk}},$$

where $(\cdot)_+$ denotes the hinge function. The algorithm for updating $\mathbf{U}$ is summarized in Algorithm 2.

4.3.2 Update of Matrix V

The update of $\mathbf{V}$ with $\mathbf{U}$ fixed is a least squares problem with ℓ_2 norm regularization. It can also be decomposed into N optimization problems, with each corresponding to one $\boldsymbol{v}_n$ and can be solved in parallel:

$$\min_{\boldsymbol{v}_n} \|\boldsymbol{d}_n - \mathbf{U}\boldsymbol{v}_n\|_2^2 + \lambda_2 \|\boldsymbol{v}_n\|_2^2,$$

for $n = 1, \cdots, N$. It is a standard ℓ_2-regularized least squares problem (also known as Ridge Regression in statistics) and the solution is:

$$\boldsymbol{v}_n^* = \left(\mathbf{U}^T\mathbf{U} + \lambda_2\mathbf{I}\right)^{-1} \mathbf{U}^T \boldsymbol{d}_n.$$

Algorithm 3 shows the procedure[2].

Algorithm 2 Update**U**

```
Require: $\mathbf{D} \in \mathbb{R}^{M\times N}$, $\mathbf{V} \in \mathbb{R}^{K\times N}$
 1: $\mathbf{S} \leftarrow \mathbf{V}\mathbf{V}^T$
 2: $\mathbf{R} \leftarrow \mathbf{D}\mathbf{V}^T$
 3: for $m = 1 : M$ do
 4:     $\bar{\boldsymbol{u}}_m \leftarrow \mathbf{0}$
 5:     repeat
 6:         for $k = 1 : K$ do
 7:             $w_{mk} \leftarrow r_{mk} - \sum_{l\neq k} s_{kl}u_{ml}$
 8:             $u_{mk} \leftarrow \frac{\left(|w_{mk}| - \frac{1}{2}\lambda_1\right)_+ \operatorname{sign}(w_{mk})}{s_{kk}}$
 9:         end for
10:     until convergence
11: end for
12: return U
```

Algorithm 3 Update**V**

```
Require: $\mathbf{D} \in \mathbb{R}^{M\times N}$, $\mathbf{U} \in \mathbb{R}^{M\times K}$
 1: $\mathbf{\Sigma} \leftarrow \left(\mathbf{U}^T\mathbf{U} + \lambda_2\mathbf{I}\right)^{-1}$
 2: $\mathbf{\Phi} \leftarrow \mathbf{U}^T\mathbf{D}$
 3: for $n = 1 : N$ do
 4:     $\boldsymbol{v}_n \leftarrow \mathbf{\Sigma}\boldsymbol{\phi}_n$, where $\boldsymbol{\phi}_n$ is the $n^{th}$ column of $\mathbf{\Phi}$
 5: end for
 6: return V
```

4.4 Implementation on MapReduce

MapReduce [9] is a computing model that supports distributed computing on large datasets. MapReduce expresses a computing task as a series of Map and Reduce operations and performs the task by executing the operations in a distributed computing environment. In this paper, we implement RLSI on MapReduce, referred to as Distributed RLSI, as shown in Figure 1. At each iteration the algorithm updates $\mathbf{U}$ and $\mathbf{V}$ using the following MapReduce operations:

Map-1 Broadcast $\mathbf{S} = \mathbf{V}\mathbf{V}^T$ and map $\mathbf{R} = \mathbf{D}\mathbf{V}^T$ on m ($m = 1, \cdots, M$) such that all of the entries in the m^{th} row of $\mathbf{R}$ are shuffled to the same machine in the form of $\langle m, \bar{\boldsymbol{r}}_m, \mathbf{S}\rangle$, where $\bar{\boldsymbol{r}}_m$ is the column vector whose entries are those of the m^{th} row of $\mathbf{R}$.

Reduce-1 Take $\langle m, \bar{\boldsymbol{r}}_m, \mathbf{S}\rangle$ and emit $\langle m, \bar{\boldsymbol{u}}_m\rangle$, where $\bar{\boldsymbol{u}}_m$ is the optimal solution for the m^{th} optimization problem (Eq. (3)). We have $\mathbf{U} = [\bar{\boldsymbol{u}}_1, \cdots, \bar{\boldsymbol{u}}_M]^T$.

Map-2 Broadcast $\mathbf{\Sigma} = \left(\mathbf{U}^T\mathbf{U} + \lambda_2\mathbf{I}\right)^{-1}$ and map $\mathbf{\Phi} = \mathbf{U}^T\mathbf{D}$ on n ($n = 1, \cdots, N$) such that the entries in the n^{th} column of $\mathbf{\Phi}$ are shuffled to the same machine in the form of $\langle n, \boldsymbol{\phi}_n, \mathbf{\Sigma}\rangle$, where $\boldsymbol{\phi}_n$ is the n^{th} column of $\mathbf{\Phi}$.

Reduce-2 Take $\langle n, \boldsymbol{\phi}_n, \mathbf{\Sigma}\rangle$ and emit $\langle n, \boldsymbol{v}_n = \mathbf{\Sigma}\boldsymbol{\phi}_n\rangle$. We have $\mathbf{V} = [\boldsymbol{v}_1, \cdots, \boldsymbol{v}_N]$.

Note that the data partitioning schemas for $\mathbf{R}$ in Map-1 and for $\mathbf{\Phi}$ in Map-2 are different. $\mathbf{R}$ is split such that entries in the same row

[2]If K is large such that the matrix inversion $\left(\mathbf{U}^T\mathbf{U} + \lambda_2\mathbf{I}\right)^{-1}$ is hard, we can employ gradient descent in the update of $\boldsymbol{v}_n$.

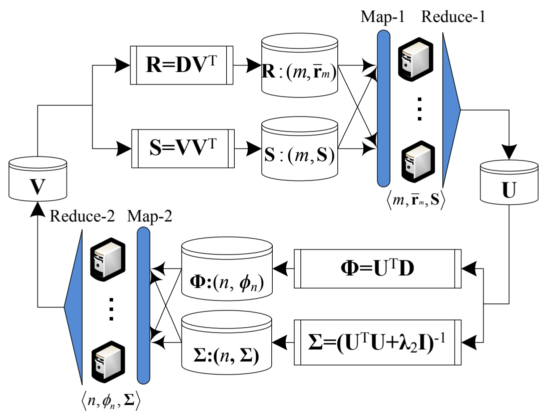

Figure 1: Update of U and V on MapReduce.

(corresponding to one term) are shuffled to the same machine while $\mathbf{\Phi}$ is split such that entries in the same column (corresponding to one document) are shuffled to the same machine.

There are a number of large scale matrix multiplication operations in operation Map-1 ($\mathbf{DV}^T$ and $\mathbf{VV}^T$) and Map-2 ($\mathbf{U}^T\mathbf{D}$ and $\mathbf{U}^T\mathbf{U}$). These matrix multiplication operations can also be conducted on MapReduce infrastructure efficiently. As example, $\mathbf{DV}^T$ can be calculated as $\sum_{n=1}^{N} \boldsymbol{d}_n\boldsymbol{v}_n^T$ and thus fully parallelized. For details please refer to [4, 20].

4.5 Discussion

We discuss the properties of RLSI with ℓ_1 norm on $\mathbf{U}$ and ℓ_2 norm on $\mathbf{V}$ as example.

4.5.1 Relationship with Other Methods

Despite having better scalability properties, RLSI is closely related to existing topic modeling methods such as LSI, PLSI, and Sparse Coding. In [30], the relationship between LSI and PLSI are discussed, from the view point of loss function and regularization. We describe their framework, so we can describe RLSI in the context of existing approaches. In that framework, topic modeling is considered as a problem of optimizing the following general loss function

$$\min_{(\mathbf{U},\mathbf{V})\in C} \quad \mathcal{B}(\mathbf{D}\|\mathbf{UV}) + \lambda\mathcal{R}(\mathbf{U},\mathbf{V}),$$

where $\mathcal{B}(\cdot\|\cdot)$ is generalized Bregman divergence with non-negative values and is equal to zero if and only if the two inputs are equivalent; $\mathcal{R}(\cdot,\cdot) \geq 0$ is the regularization on the two inputs; C is the solution space; and λ is a coefficient making trade-off between the divergence and regularization.

Different choices of $\mathcal{B}$, $\mathcal{R}$, and C lead to different topic modeling techniques. Table 2 shows the relationship between RLSI and existing methods of LSI, PLSI, and Sparse Coding. (Suppose that we first conduct normalization $\sum_{m,n} d_{mn} = 1$ in PLSI [11].) Viewing topic modeling methods in this framework, the major question is how to conduct regularization as well as optimization to make the learned topics coherent and readable.

4.5.2 Probabilistic and Non-probabilistic Models

Many non-probabilistic topic modeling techniques, such as LSI, Sparse Coding, and RLSI can be translated into a probabilistic framework, as shown in Figure 2.

In the probabilistic framework, columns of the term-topic matrix $\boldsymbol{u}_k$'s are assumed to be independent from each other and columns of the topic-document matrix $\boldsymbol{v}_n$'s are regarded as latent variables.

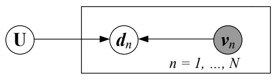

Figure 2: Probabilistic framework for non-probabilistic methods.

Table 3: Priors/constraints in different non-probabilistic methods.

Method	Prior/Constraint on $\boldsymbol{u}_k$	Prior/Constraint on $\boldsymbol{v}_n$
LSI	orthonormality	orthogonality
Sparse Coding	$\|\boldsymbol{u}_k\|_2^2 \leq 1$	$p(\boldsymbol{v}_n) \propto \exp(-\lambda\|\boldsymbol{v}_n\|_1)$
RLSI	$p(\boldsymbol{u}_k) \propto \exp(-\lambda_1\|\boldsymbol{u}_k\|_1)$	$p(\boldsymbol{v}_n) \propto \exp\left(-\lambda_2\|\boldsymbol{v}_n\|_2^2\right)$

Next, each document $\boldsymbol{d}_n$ is assumed to be generated according to a Gaussian distribution conditioned on $\mathbf{U}$ and $\boldsymbol{v}_n$, i.e., $p(\boldsymbol{d}_n|\mathbf{U},\boldsymbol{v}_n) \propto \exp\left(-\|\boldsymbol{d}_n - \mathbf{U}\boldsymbol{v}_n\|_2^2\right)$. Furthermore, all the pairs $(\boldsymbol{d}_n, \boldsymbol{v}_n)$ are conditionally independent given $\mathbf{U}$.

Different techniques use different priors or constraints on $\boldsymbol{u}_k$'s and $\boldsymbol{v}_n$'s. Table 3 lists the priors or constraints used in LSI, Sparse Coding, and RLSI, respectively. It can be shown that LSI, Sparse Coding, and RLSI can be obtained with Maximum A Posteriori (MAP) Estimation [22]. That is to say, the techniques can be understood in the same framework.

4.5.3 Scalability Comparison

As explained, several methods for improving the efficiency and scalability of existing topic models, especially LDA have been proposed. Table 4 shows the space and time complexities of AD-LDA [24], Async-CBS, Async-CVB [2], and Distributed RLSI, where AvgDL is the average document length in the collection and γ is the sparsity of topics.

The space complexity of AD-LDA (also Async-CGS and Async-CVB) is of order $\frac{N\times\text{AvgDL}+NK}{P} + MK$, where MK is for storing the term-topic matrix on each processor. For a large text collection, the vocabulary size M will be very large and thus the space complexity will be very high. This will hinder it from being applied to large datasets in real applications.

The space complexity of Distributed RLSI is $\frac{N\times\text{AvgDL}+(1+\gamma)MK+2NK}{P} + K^2$ for updating $\mathbf{U}$ and $\mathbf{V}$, where K^2 is for storing $\mathbf{S}$ or $\mathbf{\Sigma}$, $\frac{(1+\gamma)MK}{P}$ is for storing $\mathbf{U}$ and $\mathbf{R}$ in P processors, and $\frac{2NK}{P}$ is for storing $\mathbf{V}$ and $\mathbf{\Phi}$ in P processors. Since $K \ll M$, it is clear that Distributed RLSI has better scalability. We can reach the same conclusion when comparing Distributed RLSI with other parallel/distributed topic modeling methods. The key is that Distributed RLSI can distribute both terms and documents over P processors. The sparsity on the term-topic matrix can also help save the space in each processor.

The time complexities of different topic modeling methods are also listed. For Distributed RLSI, I is the number of inner iterations in Algorithm 2; T_U and T_V are for the matrix operations in Algorithms 2 and 3 (e.g., $\mathbf{VV}^T$, $\mathbf{DV}^T$, $\mathbf{U}^T\mathbf{U}$, $\mathbf{U}^T\mathbf{D}$, and matrix inversion), respectively:

$$T_U = \max\left\{\frac{\text{AvgDL}\times NK}{P} + nnz(\mathbf{R})\log P, \frac{NK^2}{P} + K^2\log P\right\},$$

$$T_V = \max\left\{\frac{\text{AvgDL}\times \gamma NK}{P} + nnz(\mathbf{\Phi})\log P, \frac{M(\gamma K)^2}{P} + K^2\log P + K^3\right\},$$

where $nnz(\cdot)$ is the number of nonzero entries in the input matrix. For details please refer to [20]. Note that the time complexities of these methods are comparable.

5. RELEVANCE RANKING

Table 2: Optimization framework for different topic modeling methods.

Method	$\mathcal{B}(\mathbf{D}\|\|\mathbf{UV})$	$\mathcal{R}(\mathbf{U},\mathbf{V})$	Constraint on $\mathbf{U}$	Constraint on $\mathbf{V}$
LSI	$\|\|\mathbf{D}-\mathbf{UV}\|\|_F^2$	—	$\mathbf{U}^T\mathbf{U}=\mathbf{I}$	$\mathbf{VV}^T=\mathbf{\Lambda}^2$ ($\mathbf{\Lambda}$ is diagonal)
PLSI	$\sum_{mn}\left(d_{mn}\log\frac{d_{mn}}{(\mathbf{UV})_{mn}}\right)$	—	$\mathbf{U}^T\mathbf{1}=\mathbf{1}$, $u_{mk}\geq 0$	$\mathbf{1}^T\mathbf{V1}=1$, $v_{kn}\geq 0$
Sparse Coding	$\|\|\mathbf{D}-\mathbf{UV}\|\|_F^2$	$\sum_n \|\|\boldsymbol{v}_n\|\|_1$	$\|\|\boldsymbol{u}_k\|\|_2^2\leq 1$	—
RLSI	$\|\|\mathbf{D}-\mathbf{UV}\|\|_F^2$	$\sum_k \|\|\boldsymbol{u}_k\|\|_1$, $\sum_n \|\|\boldsymbol{v}_n\|\|_2^2$	—	—

Table 4: Complexity of parallel/distributed topic models.

Method	Space complexity	Time complexity (per iter)
AD-LDA	$\frac{N\times\text{AvgDL}+NK}{P}+MK$	$\frac{NK\times\text{AvgDL}}{P}+MK\log P$
Async-CGS	$\frac{N\times\text{AvgDL}+NK}{P}+2MK$	$\frac{NK\times\text{AvgDL}}{P}+MK\log P$
Async-CVB	$\frac{N\times\text{AvgDL}+2NK}{P}+4MK$	$\frac{MK}{P}+MK\log P$
Distributed RLSI	$\frac{N\times\text{AvgDL}+(1+\gamma)MK+2NK}{P}+K^2$	$\frac{IMK^2+NK^2}{P}+T_U+T_V$

Topic models can be used in a wide variety of applications. We apply RLSI to relevance ranking in information retrieval (IR) and evaluate its performance in comparison to existing topic modeling methods. The use of topic modeling techniques such as LSI was proposed in IR many years ago [10]. A more recent paper [35] demonstrated improvements in retrieval performance by applying topic modeling on modern test collections. We do not replicate their precise ranking approach here, since it relies on a probabilistic topic model, but we achieve similar gains.

The advantage of incorporating topic modeling in relevance ranking is to reduce "term mismatch". Traditional relevance models, such as VSM [29] and BM25 [27], are all based on term matching. The term mismatch problem arises when the authors of documents and the users of search system use different terms to describe the same concepts, and thus relevant documents get low relevance scores. For example, if a query contains the term "airplane" but a relevant document instead contains the term "aircraft", then there is a mismatch and the document may not be easily distinguished from an irrelevant one. In the topic space, however, it is very likely that the two terms are in the same topic, and thus the use of matching score in the topic space can help improve relevance ranking. In practice it is beneficial to combine topic matching scores with term matching scores, to leverage both broad topic matching and specific term matching.

To do so, given a query and document, we must calculate their matching scores in both term space and topic space. For query q, we represent it in the topic space:

$$\boldsymbol{v}_q = \arg\min_{\boldsymbol{v}} \|\boldsymbol{q}-\mathbf{U}\boldsymbol{v}\|_2^2 + \lambda_2\|\boldsymbol{v}\|_2^2,$$

where vector $\boldsymbol{q}$ is the tf-idf representation of query q in the term space[3]. Similarly, for document d (and its tf-idf representation $\boldsymbol{d}$ in the term space) we represent it in the topic space as $\boldsymbol{v}_d$. The matching score between the query and the document in the topic space is, then, calculated as the cosine similarity between $\boldsymbol{v}_q$ and $\boldsymbol{v}_d$:

$$s_{topic}(q,d) = \frac{\langle \boldsymbol{v}_q, \boldsymbol{v}_d\rangle}{\|\boldsymbol{v}_q\|_2 \cdot \|\boldsymbol{v}_d\|_2}. \quad (4)$$

The topic matching score $s_{topic}(q,d)$ is combined with the conventional term matching score $s_{term}(q,d)$, for final relevance ranking. There are several ways to conduct the combination. A simple and effective approach is to use a linear combination. The final relevance ranking score $s(q,d)$ is:

$$s(q,d) = \alpha s_{topic}(q,d) + (1-\alpha)s_{term}(q,d), \quad (5)$$

where $\alpha \in [0,1]$ is the coefficient. $s_{term}(q,d)$ can be calculated with any of the conventional relevance models such as VSM and BM25.

Another combination approach is to incorporate the topic matching score as a feature in a learning to rank model, e.g., LambdaRank [5]. In this paper, we use both approaches in our experiments.

6. EXPERIMENTS

We have conducted experiments to compare different RLSI regularization strategies, to compare RLSI with existing methods, and to test scalability and retrieval performance of RLSI using several datasets.

6.1 Experimental Settings

Our three TREC datasets were AP, WSJ, and OHSUMED, which have been widely used in relevance ranking experiments. We also used a large real-world web dataset from a commercial web search engine, containing about 1.6 million documents and 10 thousand queries. Each dataset consists of a document collection, a set of queries, and relevance judgments on some documents with respect to each query. For all four datasets, only the retrieved documents were included and a standard list of stop words was removed. For the Web dataset, we further discarded the terms whose frequencies in the whole dataset are less than two. Table 5 gives some statistics on the datasets.

In AP and WSJ the relevance judgments are at two levels: "relevant" or "irrelevant". In OHSUMED, the relevance judgments are at three levels: "definitely relevant", "partially relevant", and "not relevant". In the Web dataset, there are five levels: "perfect", "excellent", "good", "fair", and "bad". In the experiments of retrieval performance, we used MAP and NDCG at the positions of 1, 3, 5, and 10 for evaluating retrieval performance. In calculating MAP, we consider "definitely relevant" and "partially relevant" in OHSUMED, and "perfect", "excellent", and "good" in Web dataset as "relevant".

In the experiments on TREC datasets (Section 6.2 and Section 6.3), no validation set was used since we only had small query sets, making it difficult to hold out a validation set of meaningful size in each case. Instead, we chose to evaluate each model in a predefined grid of parameters, showing its performance under the best parameter choices. In the experiments on the Web dateset (Section 6.4), the queries were randomly split into training/validation/test sets, with 6000/2000/2680 queries, respectively. We trained the ranking models with the training set, selected the best models with the validation set, and evaluated the performances of the methods with the test set.

The experiments on AP, WSJ, and OHSUMED were conducted on a server with Intel Xeon 2.33GHZ CPU, 16GB RAM. The experiments on the Web dataset were conducted on a distributed system and the Distributed RLSI was implemented with SCOPE language [6].

6.2 Regularization in RLSI

Our comparison of different RLSI regularization strategies was carried out on AP, WSJ, and OHSUMED datasets. Regulariza-

[3]Using $\boldsymbol{v}_q = \arg\min_{\boldsymbol{v}} \|\boldsymbol{q}-\mathbf{U}\boldsymbol{v}\|_2^2+\lambda_2\|\boldsymbol{v}\|_1$ if ℓ_1 norm is imposed on $\mathbf{V}$

Table 5: Dataset statistics.

Dataset	AP	WSJ	OHSUMED	Web dataset
# terms	83,541	106,029	26,457	7,014,881
# documents	29,528	45,305	14,430	1,562,807
# queries	250	250	106	10,680

tion on **U** and **V** via either ℓ_1 or ℓ_2 norm gives us four RLSI variants: RLSI (Uℓ_1-Vℓ_2), RLSI (Uℓ_2-Vℓ_1), RLSI (Uℓ_1-Vℓ_1), and RLSI (Uℓ_2-Vℓ_2), where RLSI (Uℓ_1-Vℓ_2) means, for example, applying ℓ_1 norm on **U** and ℓ_2 norm on **V**. Parameters K, λ_1, λ_2, and α were respectively set in ranges of $[10, 50]$, $[0.01, 1]$, $[0.01, 1]$, and $[0.1, 1]$ for all variants.

We first compared the RLSI variants in terms of topic readability, by looking at the contents of topics they generated. As example, Table 6 shows 10 topics (randomly selected) and the average topic compactness (AvgComp) on AP dataset, for all four RLSI variants, when $K = 20$ and λ_1 and λ_2 are the optimal parameters for the retrieval experiment described next. Here, average topic compactness is defined as average ratio of terms with non-zero weights per topic. For each topic, its top 5 weighted terms are shown. From the results, we have found that 1) If ℓ_1 norm is imposed on either **U** or **V**, RLSI can always discover readable topics; 2) Without ℓ_1 norm regularization (i.e., RLSI(Uℓ_2-Vℓ_2)), many topics are not readable; 3) If ℓ_1 norm is only imposed on **V** (i.e. RLSI (Uℓ_2-Vℓ_1)), then the discovered topics are not compact or sparse (e.g., AvgComp = 1). We also conducted the same experiments on WSJ and OHSUMED and observed similar phenomena. The examining topics on them are not shown due to space limitation.

We also compared the RLSI variants in terms of retrieval performance. Specifically, for each of the RLSI variants, we combined topic matching scores ($s_{topic}(q, d)$ in Eq. (5)) with term matching scores given by conventional IR models of VSM or BM25. Since BM25 performed better than VSM on AP and WSJ, and VSM performed better than BM25 on OHSUMED, we combined the topic matching scores with BM25 on AP and WSJ, and with VSM on OHSUMED. The methods we tested were denoted as "BM25+RLSI (Uℓ_1-Vℓ_2)", "BM25+RLSI (Uℓ_2-Vℓ_1)", "BM25+RLSI (Uℓ_1-Vℓ_1)", "BM25+RLSI (Uℓ_2-Vℓ_2)", etc. Figures 3, 4, and 5 show the retrieval performance of RLSI variants achieved by the best parameter setting on AP, WSJ, and OHSUMED, respectively. From the results, we can see that 1) All of these methods can improve over the baseline and in most cases the improvement is statistically significant (t-test, p-value < 0.05); 2) Among the RLSI variants, RLSI (Uℓ_1-Vℓ_2) performs best and RLSI (Uℓ_2-Vℓ_2) performs worst.

Table 7 summarizes the experimental results in terms of topic readability, topic compactness, and retrieval performance. From the result, we can see that in RLSI, ℓ_1 norm regularization is essential for discovering readable topics, and the discovered topics will also be compact if ℓ_1 norm is imposed on **U**. Furthermore, between the two RLSI variants with good topic readability and compactness, i.e., RLSI (Uℓ_1-Vℓ_2) and RLSI (Uℓ_1-Vℓ_1), RLSI (Uℓ_1-Vℓ_2) performs better in improving retrieval performance. Thus we conclude that it is a better practice to apply ℓ_1 norm on **U** and ℓ_2 norm on **V** in RLSI, for achieving good topic readability, topic compactness, and retrieval performance.

We will use RLSI (Uℓ_1-Vℓ_2) in the following experiments and denote it as RLSI for simplicity.

6.3 Comparison of Topic Models

In this experiment, we compared RLSI with LDA, PLSI, and LSI on AP, WSJ, and OHSUMED datasets.

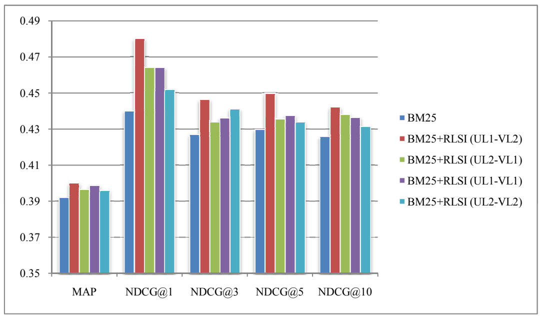

Figure 3: Retrieval performance of RLSI variants on AP.

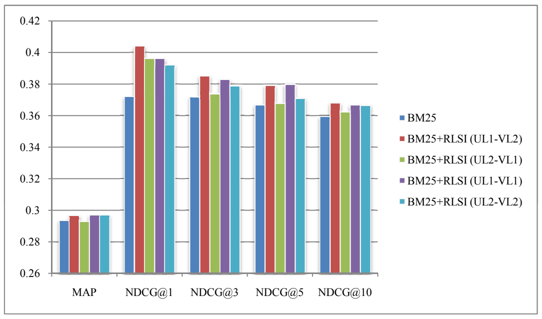

Figure 4: Retrieval performance of RLSI variants on WSJ.

We first compared RLSI with LDA, PLSI, and LSI in terms of topic readability, by looking at the topics they generated. We made use of the tools available on Internet for creating the baselines[4]. The number of topics K was again set to 20 for all the methods. In RLSI, λ_1 and λ_2 were the optimal parameters used in Section 6.2 (i.e., $\lambda_1 = 0.5$ and $\lambda_2 = 1.0$). For LDA, PLSI, and LSI, there was no additional parameter to tune.

Table 8 shows 10 randomly selected topics discovered by RLSI, LDA, PLSI, and LSI and the average topic compactness (AvgComp) on AP dataset. For each topic, its top 5 weighted terms are shown. From the results, we have found 1) RLSI can discover readable and compact (e.g., AvgComp = 0.0075) topics; 2) PLSI and LDA can discover coherent and readable topics as expected, however the discovered topics are not compact (e.g., AvgComp = 0.9534 and AvgComp = 1, respectively); 3) LDA performs better than PLSI. There is some redundancy in the topics discovered by PLSI; 4) The topics discovered by LSI were hard to understand, and this may be due to its orthogonality assumption. We also conducted the same experiments on WSJ and OHSUMED and observed similar phenomena. The results on them are not shown due to space limitation.

We also tested the performance of RLSI in terms of retrieval performance, in comparison to LSI, PLSI, LDA. The experimental settings was similar to that of used in Section 6.2. Parameters K and α were respectively set in ranges of $[10, 50]$ and $[0.1, 1]$ for all four methods, and parameters λ_1 and λ_2 in RLSI were respectively set in ranges of $[0.01, 1]$ and $[0.01, 1]$. Figures 6, 7, and 8 show retrieval performance achieved by the best parameter setting on AP, WSJ, and OHSUMED, respectively. From the results, we can see that RLSI can *significantly* improve the baseline (t-test, p-value < 0.05), going beyond the simple term matching paradigm. Among the different topic modeling methods, RLSI performs slightly better than

[4]LSI: `http://tedlab.mit.edu/~dr/SVDLIBC/`; PLSI: `http://www.lemurproject.org/`; LDA: `http://www.cs.princeton.edu/~blei/lda-c/`

Table 6: Topics discovered by RLSI variants from AP dataset.

	Topic 1	Topic 2	Topic 3	Topic 4	Topic 5	Topic 6	Topic 7	Topic 8	Topic 9	Topic 10
	opec	africa	aid	school	noriega	percent	plane	israeli	nuclear	bush
	oil	south	virus	student	panama	billion	crash	palestinian	soviet	dukakis
RLSI ($U\ell_1$-$V\ell_2$)	cent	african	infect	teacher	panamanian	rate	flight	israel	treaty	campaign
AvgComp = 0.0075	barrel	angola	test	educate	delval	0	air	arab	missile	quayle
	price	apartheid	patient	college	canal	trade	airline	plo	weapon	bentsen
	nuclear	court	noriega	africa	cent	israeli	dukakis	student	plane	percent
	treaty	judge	panama	south	opec	palestinian	bush	school	crash	billion
RLSI ($U\ell_2$-$V\ell_1$)	missile	prison	panamanian	african	oil	israel	jackson	teacher	flight	rate
AvgComp = 1	weapon	trial	delval	angola	barrel	arab	democrat	educate	air	0
	soviet	sentence	canal	apartheid	price	plo	campaign	college	airline	trade
	court	plane	dukakis	israeli	africa	soviet	school	yen	cent	noriega
	prison	crash	bush	palestinian	south	treaty	student	trade	opec	panama
RLSI ($U\ell_1$-$V\ell_1$)	judge	air	jackson	israel	african	missile	teacher	dollar	oil	panamanian
AvgComp = 0.0197	sentence	flight	democrat	arab	angola	nuclear	educate	market	barrel	delval
	trial	airline	campaign	plo	apartheid	gorbachev	college	japan	price	canal
	dukakis	palestinian	soviet	school	africa	dukakis	soviet	drug	percent	soviet
	oil	israeli	noriega	student	south	bush	treaty	cent	billion	israeli
RLSI ($U\ell_2$-$V\ell_2$)	opec	israel	panama	bakker	iran	democrat	student	police	price	missile
AvgComp = 1	cent	arab	drug	trade	african	air	nuclear	student	trade	israel
	bush	plo	quake	china	dukakis	jackson	missile	percent	cent	treaty

Table 8: Topics discovered by RLSI, LDA, PLSI, and LSI from AP dataset.

	Topic 1	Topic 2	Topic 3	Topic 4	Topic 5	Topic 6	Topic 7	Topic 8	Topic 9	Topic 10
	opec	africa	aid	school	noriega	percent	plane	israeli	nuclear	bush
	oil	south	virus	student	panama	billion	crash	palestinian	soviet	dukakis
RLSI	cent	african	infect	teacher	panamanian	rate	flight	israel	treaty	campaign
AvgComp = 0.0075	barrel	angola	test	educate	delval	0	air	arab	missile	quayle
	price	apartheid	patient	college	canal	trade	airline	plo	weapon	bentsen
	soviet	school	dukakis	party	year	water	price	court	air	iran
	nuclear	student	democrat	govern	new	year	year	charge	plane	iranian
LDA	union	year	campaign	minister	time	fish	market	case	flight	ship
AvgComp = 1	state	educate	bush	elect	television	animal	trade	judge	crash	iraq
	treaty	university	jackson	nation	film	0	percent	attorney	airline	navy
	company	israeli	year	year	bush	court	soviet	year	plane	year
	million	iran	state	state	dukakis	charge	treaty	state	flight	state
PLSI	share	israel	new	new	democrat	attorney	missile	new	airline	new
AvgComp = 0.9534	billion	palestinian	nation	nation	campaign	judge	nuclear	nation	crash	people
	stock	arab	govern	0	republican	trial	gorbachev	govern	air	nation
	soviet	567	0	earthquake	drug	0	israel	yen	urgent	student
	percent	234	yen	quake	school	dukakis	israeli	dukakis	oil	school
LSI	police	0	dollar	richter	test	bush	student	bush	opec	noriega
AvgComp = 1	govern	percent	percent	scale	court	jackson	palestinian	dollar	dukakis	panama
	state	12	tokyo	damage	dukakis	dem	africa	jackson	cent	teacher

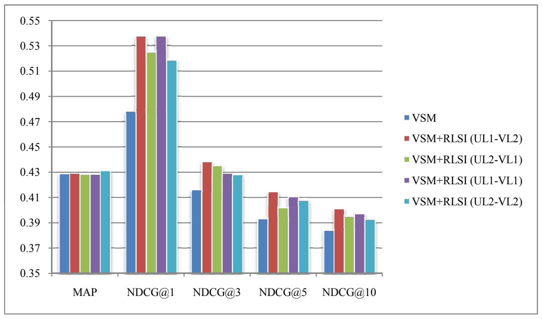

Figure 5: Retrieval performance of RLSI variants on OHSUMED.

Table 7: Performance of the RLSI variants.

	Readability	Compactness	Retrieval performance
RLSI ($U\ell_1$-$V\ell_2$)	√	√	√
RLSI ($U\ell_2$-$V\ell_1$)	√	×	×
RLSI ($U\ell_1$-$V\ell_1$)	√	√	×
RLSI ($U\ell_2$-$V\ell_2$)	×	×	×

the other methods, and sometimes the improvements are statistically significant (t-test, p-value < 0.05). We conclude that RLSI is a proper choice for combining topic matching and term matching.

6.4 Experiment on Web Dataset

We tested the scalability of RLSI using a large real-world web dataset. Table 9 lists the sizes of popular datasets used to evaluate existing distributed/parallel topic models, as well as the size of our Web dataset. We can see that the number of terms in Web dataset is much larger (about 35 times of the number of terms in Wiki-200T), which hinders the scaling up of existing parallel/distributed topic models, as they need to keep the dense term-topic matrix in memory on each processor. Distributed RLSI, on the other hand, can distribute the terms and documents over processors and thus can handle the Web dataset effectively and efficiently. (Note that it is difficult for us to re-implement existing parallel topic modeling methods, because most of them require special computing infras-

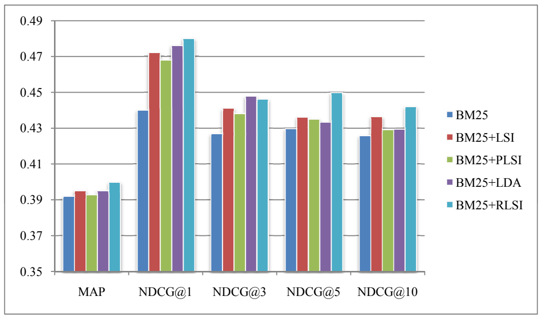

Figure 6: Retrieval performance on AP.

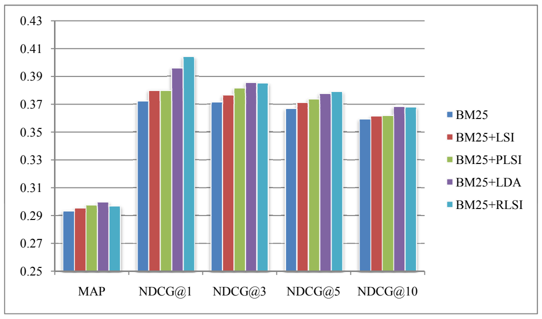

Figure 7: Retrieval performance on WSJ.

tructures and the development costs of the methods are high.)

In our experiments, the number of topics K was set to 500, λ_1 and λ_2 were again set to 0.5 and 1.0 respectively. It took about 1.5 hours for Distributed RLSI to complete an iteration on the MapReduce system with 16 processors. Table 10 shows 10 randomly sampled topics and the overall topic compactness on the Web dataset. We can see that the topics obtained by RLSI are compact and readable.

Next, we tested retrieval performance of Distributed RLSI. We randomly split the queries into training/validation/test sets, with 6000/2000/2680 queries, respectively. We took LambdaRank [5] as the baseline. There are 16 features used in the LambdaRank model, including BM25, PageRank, and Query-Exact-Match. In our methods, the topic matching scores by RLSI were used as a new feature in LambdaRank, denoted as "LambdaRank+RLSI". Figure 9 shows the results on the test set, indicating that topics discovered by RLSI allowed "LambdaRank+RLSI" to significantly (t-test, p-value < 0.01) outperform the baseline method of LambdaRank.

Finally, since other papers reduced input vocabulary size, we tested the effect of reducing the vocabulary size in RLSI. Specifically, we removed the terms whose total term frequency is less than 100 from the Web dataset obtaining a new dataset with 222,904 terms. We applied RLSI on the new dataset with parameters $K = 500, \lambda_1 = 0.5$ and $\lambda_2 = 1.0$. We then created a LambdaRank model with topic matching scores as a feature, denoted as "LambdaRank+RLSI (Reduced Vocabulary)". Figure 9 shows the retrieval performance of "LambdaRank+RLSI (Reduced Vocabulary)" on the test set. The result indicates that reducing the vocabulary size will sacrifice learning accuracy of RLSI and consequently hurt the retrieval performance. We conducted t-tests on the differences between "LambdaRank+RLSI (Reduced Vocabulary)" and "LambdaRank+RLSI" and found that the difference is statistically significant (p-value < 0.01). We observed the same trends on the TREC datasets for RLSI and LDA, but we do not report the details due to space limitation.

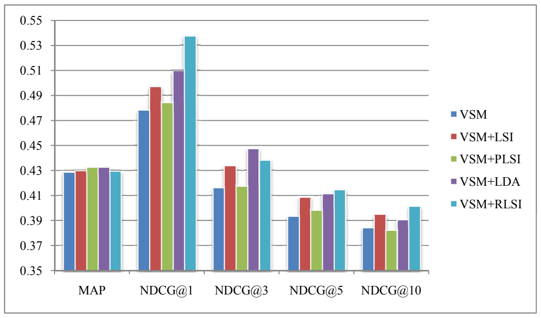

Figure 8: Retrieval performance on OHSUMED.

Table 9: Size of datasets.

Dataset	# docs	# terms	Applied algorithms
NIPS	1,500	12,419	Async-CVB, Async-CGS, PLDA
Wiki-200T	2,122,618	200,000	PLDA+
PubMed	8,200,000	141,043	AD-LDA, Async-CVB, Async-CGS
Web dataset	1,562,807	7,014,881	Distributed RLSI

7. CONCLUSIONS

In this paper, we have studied topic modeling from the viewpoint of enhancing scalability and retrieval performance. We have proposed a new method for topic modeling, called Regularized Latent Semantic Indexing (RLSI). RLSI formalizes topic modeling as minimization of a quadratic loss function with a regularization (either ℓ_1 or ℓ_2 norm). Although similar techniques have been used in other fields, such as sparse coding in computer vision, this is the first comprehensive study of regularization for topic modeling, as far as we know. It is exactly the formulation of RLSI that makes its optimization process decomposable, and thus scalable. Specifically, RLSI replaces the orthogonality constraint or probability distribution constraints with regularization. Therefore, RLSI can be more easily implemented in a parallel and/or distributed computing environment, such as MapReduce. We presented a specific algorithm for running RLSI on MapReduce.

In our experiments we tested different variants of RLSI and confirmed that the sparse topic regularization and smooth document regularization is the best choice from the viewpoint of overall performance. Specifically the ℓ_1 norm on topics (making topics sparse) and ℓ_2 norm on document representations gave the best readability and retrieval performance.

Experimental results on TREC data and large scale web data show that RLSI is better than or comparable with existing methods such as LSI, PLSI, and LDA in terms of readability of topics and accuracy in relevance ranking. We have also demonstrated that RLSI can scale up to large document collection with 1.6 million documents and 7 million terms, which is very difficult for exiting methods. Most previous papers reduced the input vocabulary size to tens of thousands of terms. As far as we know, this is the largest size which the topic modeling methods can handle so far. We have also verified that RLSI can help improve web search relevance.

As future work, we plan to further enhance the scale of experiments to process even larger datasets. We also want to further study the theoretical properties of RLSI and new applications of RLSI.

8. REFERENCES

[1] L. AlSumait, D. Barbara, and C. Domeniconi. On-line lda: Adaptive topic models for mining text streams with applications to topic detection and tracking. In *ICDM*, 2008.

Table 10: Topics discovered by RLSI from Web dataset (AvgComp = 0.0035).

Topic 1	Topic 2	Topic 3	Topic 4	Topic 5	Topic 6	Topic 7	Topic 8	Topic 9	Topic 10
casino	mortgage	wheel	cheap	login	christian	google	obj	spywar	friend
poker	loan	rim	flight	password	bible	web	pdf	anti	myspace
slot	credit	tire	hotel	username	church	yahoo	endobj	sun	music
game	estate	truck	student	registration	god	host	stream	virus	comment
vegas	bank	car	travel	email	jesus	domain	xref	adwar	photo

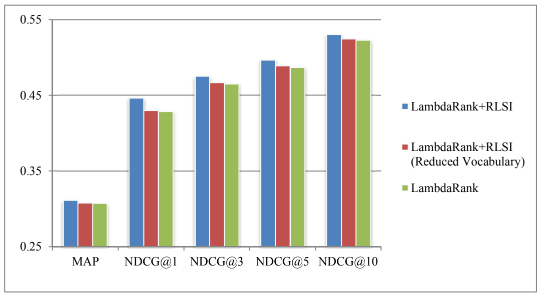

Figure 9: Retrieval performance on Web dataset.

[2] A. Asuncion, P. Smyth, and M. Welling. Asynchronous distributed estimation of topic models for document analysis. *Statistical Methodology*, 2011.

[3] D. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *JMLR*, 3:993–1022, 2003.

[4] A. Buluc and J. R. Gilbert. Challenges and advances in parallel sparse matrix-matrix multiplication. In *ICPP*, pages 503–510, 2008.

[5] C. J. Burges, R. Ragno, and Q. V. Le. Learning to rank with nonsmooth cost functions. In *NIPS 19*, 2007.

[6] R. Chaiken, B. Jenkins, P.-A. Larson, B. Ramsey, D. Shakib, S. Weaver, and J. Zhou. Scope: Easy and efficient parallel processing of massive data sets. *VLDB Endow.*, 1:1265–1276, 2008.

[7] S. S. Chen, D. L. Donoho, and M. A. Saunders. Atomic decomposition by basis pursuit. *SISC*, 20:33–61, 1998.

[8] X. Chen, B. Bai, Y. Qi, Q. Lin, and J. Carbonell. Sparse latent semantic analysis. In *NIPS Workshop*, 2010.

[9] J. Dean, S. Ghemawat, and G. Inc. Mapreduce: simplified data processing on large clusters. In *OSDI*, 2004.

[10] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman. Indexing by latent semantic analysis. *J AM SOC INFORM SCI*, 41:391–407, 1990.

[11] C. Ding, T. Li, and W. Peng. On the equivalence between non-negative matrix factorization and probabilistic latent semantic indexing semantic indexing. *COMPUT STAT DATA AN*, 52:3913–3927, 2008.

[12] B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani. Least angle regression. *ANN STAT*, 32:407–499, 2004.

[13] J. Friedman, T. Hastie, H. Hofling, and R. Tibshirani. Pathwise coordinate optimization. *ANN APPL STAT*, 1:302–332, 2007.

[14] W. J. Fu. Penalized regressions: The bridge versus the lasso. *J COMPUT GRAPH STAT*, 7:397–416, 1998.

[15] M. D. Hoffman, D. M. Blei, and F. Bach. Online learning for latent dirichlet allocation. In *NIPS*, 2010.

[16] T. Hofmann. Probabilistic latent semantic indexing. In *SIGIR*, pages 50–57, 1999.

[17] D. D. Lee and H. S. Seung. Learning the parts of objects with nonnegative matrix factorization. *Nature*, 401:391–407, 1999.

[18] D. D. Lee and H. S. Seung. Algorithms for non-negative matrix factorization. In *NIPS 13*, pages 556–562. 2001.

[19] H. Lee, A. Battle, R. Raina, and A. Y. Ng. Efficient sparse coding algorithms. In *NIPS*, pages 801–808. 2007.

[20] C. Liu, H. chih Yang, J. Fan, L.-W. He, and Y.-M. Wang. Distributed nonnegative matrix factorization for web-scale dyadic data analysis on mapreduce. In *WWW*, pages 681–690, 2010.

[21] Z. Liu, Y. Zhang, and E. Y. Chang. Plda+: Parallel latent dirichlet allocation with data placement and pipeline processing. In *TIST*, 2010.

[22] J. Mairal, F. Bach, J. Ponce, G. Sapiro, and A. Zisserman. Supervised dictionary learning. In *NIPS 21*, pages 1033–1040. 2009.

[23] D. M. Mimno and McCallum. Organizing the oca: Learning faceted subjects from a library of digital books. In *JCDL*, pages 376–385, 2007.

[24] D. Newman, A. Asuncion, P. Smyth, and M. Welling. Distributed inference for latent dirichlet allocation. In *NIPS*, 2008.

[25] B. A. Olshausen and D. J. Fieldt. Sparse coding with an overcomplete basis set: a strategy employed by v1. *VISION RES*, 37:3311–3325, 1997.

[26] M. Osborne, B. Presnell, and B. Turlach. A new approach to variable selection in least squares problems. *IMA J NUMER ANAL*, 2000.

[27] S. E. Robertson, S. Walker, S. Jones, M. Hancock-Beaulieu, and M. Gatford. Okapi at trec-3. In *TREC'3*, 1994.

[28] R. Rubinstein, M. Zibulevsky, and M. Elad. Double sparsity: Learning sparse dictionaries for sparse signal approximation. *IEEE T SIGNAL PROCES*, pages 1553–1564, 2008.

[29] G. Salton, A. Wong, and C. S. Yang. A vector space model for automatic indexing. *Commun. ACM*, 18:613–620, 1975.

[30] A. P. Singh and G. J. Gordon. A unified view of matrix factorization models. In *ECMLPKDD*, pages 358–373, 2008.

[31] A. Smola and S. Narayanamurthy. An architecture for parallel topic models. *Proc. VLDB Endow.*, 3:703–710, 2010.

[32] R. Thakur and R. Rabenseifner. Optimization of collective communication operations in mpich. *INT J HIGH PERFORM C*, 19:49–66, 2005.

[33] C. Wang and D. M. Blei. Decoupling sparsity and smoothness in the discrete hierachical dirichlet process. In *NIPS*, 2009.

[34] Y. Wang, H. Bai, M. Stanton, W. yen Chen, and E. Y. Chang. Plda: Parallel latent dirichlet allocation for large-scale applications. In *AAIM*, pages 301–314, 2009.

[35] X. Wei and B. W. Croft. Lda-based document models for ad-hoc retrieval. In *SIGIR*, pages 178–185, 2006.

[36] F. Yan, N. Xu, and Y. A. Qi. Parallel inference for latent dirichlet allocation on graphics processing units. In *NIPS*, pages 2134–2142, 2009.