Topic Modeling in Semantic Space with Keywords

Xiaojia Pu[1], Rong Jin[2,3], Gangshan Wu[1], Dingyi Han[3], Gui-Rong Xue[3]
[1]State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing, China
[2]Department of Computer Science and Engineering, Michigan State University, East Lansing, USA
[3]Alibaba Group, Hangzhou, China
puxiaojia@gmail.com, gswu@nju.edu.cn,
{jinrong.jr,dingyi.han,grxue}@alibaba-inc.com

ABSTRACT

A common and convenient approach for user to describe his information need is to provide a set of keywords. Therefore, the technique to understand the need becomes crucial. In this paper, for the information need about a topic or category, we propose a novel method called TDCS(Topic Distilling with Compressive Sensing) for explicit and accurate modeling the topic implied by several keywords. The task is transformed as a topic reconstruction problem in the semantic space with a reasonable intuition that the topic is sparse in the semantic space. The latent semantic space could be mined from documents via unsupervised methods, e.g. LSI. Compressive sensing is leveraged to obtain a sparse representation from only a few keywords. In order to make the distilled topic more robust, an iterative learning approach is adopted. The experiment results show the effectiveness of our method. Moreover, with only a few semantic concepts remained for the topic, our method is efficient for subsequent text mining tasks.

Categories and Subject Descriptors

I.2.7 [**Artificial Intelligence**]: Natural Language Processing—*Text Analysis*; H.3.3 [**INFORMATION STORAGE AND RETRIEVAL**]: Information Search and Retrieval

Keywords

text mining, semantic space, compressive sensing

1. INTRODUCTION

With the rapid growth of Web data, there is an increasing need to get information on some interesting topics or categories. Due to the volume and velocity of Web data, it becomes crucial to provide an efficient and convenient way for users to get the information.

The interesting topic is often dynamic and differs a lot for various users. For example, one may be interested in 'food', while the other one is interested in 'sports'. It could be different in granularity. For example, compared with 'soccer', 'sports' is more coarse and 'world cup' is more fine. Moreover, the topic is often dynamic, which is context dependent. Therefore, it's almost impossible to define topic's categories and organize them in advance. It's essential to provide a flexible approach for user to acquire the information.

A common and convenient approach is that users describe the topic with several keywords, then an information system tries to understand the request and provide corresponding information service. The more accurately users describe the topic, the more precise service the information system could provide. In the usual manner of people's expressions, the system usually requires sufficient information to supply good service. However, in practice, a series of keywords are possible for indicating a topic if the user really has a clear information need. For example, the keywords, 'car, automobile, engine, sedan, bmw' probably imply an interest on the topic 'auto'. It's particularly attractive for the prompt need, since humans can quickly give keywords.

In such settings, given several keywords, the accurate understanding and modeling of the topic becomes an important and essential problem. The problem is not that easy, because sometimes the provided keywords are not sufficient. The complexity of the text representation and the randomness of provided keywords also increase the difficulty.

To address the problem, there are two kinds of existing methods. One is to tackle in the word space, such as bag of words(BOW) based method, e.g. VSM. The other kind of methods is to tackle in the semantic space.

VSM[23] is a purely term based method, in which the topic is represented as a bag of words, accounting for the number of occurrences of each term. This method is not sufficient for the problem, since there are many other words also beneficial to the topic.

Different with VSM, some methods try to exploit the semantic concepts or topics behind the documents or words, e.g. LSI, PLSI, LDA. Take example for LSI, through matrix decomposition, it builds a semantic space which, by intent, represents the semantic concepts in the dataset, and then projects both documents and words into the semantic dimensional space[6]. The most simple and natural approach is to model the topic in the semantic space by a projection from word space. However, it's still not accurate enough. Essentially, these methods construct semantic concepts by the exploitation of correlations among the co-occurring words. For a word, even it's strongly relevant with a topic, it's very likely to appear with the words

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from Permissions@acm.org.
CIKM'15, October 19–23, 2015, Melbourne, Australia.
© 2015 ACM. ISBN 978-1-4503-3794-6/15/10 ...$15.00.
DOI: http://dx.doi.org/10.1145/2806416.2806584.

mainly about other topics. Thus, the simple projection will inevitably introduce irrelevant semantic concepts which affects the precision of the topic.

Usually, a well specified topic should be cohesive in the semantic space[13][5]. In other words, the topic is sparse in the semantic space, which makes it possible to approximate the real semantic distribution of a topic with a few semantic concepts. In practice, utilizing only a few components of the semantic space to represent a topic could also lead to the computational efficiency in the following tasks, which is very beneficial especially for the big data. This is another important issue to motivate us to build a sparse model.

For our topic modeling problem, we try to construct a sparse topic in the semantic space with core semantic concepts. It's challenging because the number of keywords is too small. To this end, we propose a novel method called TDCS(Topic Distilling with Compressive Sensing), which aims to distill the topic in the semantic space from the keywords. The theory of compressive sensing(CS)[7][8] demonstrates that many natural signals are sparse or compressible in a transformed basis. Thus, we take advantage of CS to establish a sparse solution in the semantic space from only a few keywords. To make the result more robust, an iterative learning approach is adopted. Our proposed method is simple yet effective. Moreover, with only a few semantic concepts remained for the topic representation, our method is efficient for subsequent tasks, e.g. text classification.

The contributions of our paper can be summarized as follows:

1) To the best of our knowledge, we are the first one to use CS for topic modeling. We propose an effective method to distill the topic from only a few keywords which exploits and utilizes the sparsity of the semantic space.

2) Our method is very efficient, with less semantic components remained for topic representation, it can significantly reduce computational complexity for subsequent tasks.

3) Our method is flexible, which could be easily extended into other semantic spaces.

The rest of this paper is organized as follows. Firstly, the preliminaries are introduced in section 2. Then, our proposed method will be are described in detail in section 3 and 4. The experiments and results in section 5 demonstrate the effectiveness of our method. Some related works are discussed in section 6. Finally, we conclude our paper in section 7.

2. PRELIMINARIES

In this section, we give introduction on LSI and compressive sensing. Though we take LSI as the method for build the semantic space, our method could be easily extended into the semantic space built by other methods.

2.1 Latent Semantic Indexing

Latent semantic indexing(LSI) uses a standard matrix factorization technique, i.e. single value decomposition(SVD) to construct a latent semantic space.

Given the document-term matrix X of a corpus D, the d-th row X_d represents a document d in the corpus D. Supposing D is corpus of M documents, indexed by d. There are W distinct terms in the vocabulary.

Let the SVD of X be $X = U\Sigma V^T$, where the matrix U and V are orthogonal, and Σ is diagonal. The values $\sigma_1, \sigma_2, \ldots, \sigma_{min}\{W, M\}$ are the singular values of X.

For LSI, the matrix X is approximated by a rank-K approximation $\hat{X}$. This is usually done with a partial SVD using the singular vectors corresponding to the K largest singular values[1].

$$\hat{X} = \hat{U}\hat{\Sigma}\hat{V}^T \tag{1}$$

We can observe from Equation 1 that each term can be represented by the K-dimensional vector $\hat{T}_v$ which is the v-th row of the matrix $\hat{V}$. Similarly, the document also can be represented by the K-dimensional vector. Thus, LSI projects both terms and documents into a K-dimensional latent semantic space which could be utilized for some tasks, e.g. information retrieval.

Supposing we are given a query q which contains several terms, the query could be viewed as a short document and be projected into the latent semantic space using

$$\hat{q} = \Sigma^{-1}\hat{V}^T q. \tag{2}$$

From the practical perspective, LSI persists the principle semantic concepts in the dataset.

2.2 Compressive Sensing

Compressive sensing(CS) is an emerging technique to efficiently acquire and reconstruct a signal from a small number of non-adaptive linear measurements. The number of measurements are usually smaller than the dimension of the signal. [7][3].

The theory of CS is based on the observation that many natural signals are sparse or compressible. In other words, they have concise representations or structures when expressed in a transformed basis.

More precisely, as shown in Figure 1, let $x \in R^L$ denote a target signal in compressible basis Ψ, and $y \in R^n$ denote $n < L$ measurements of x, where $\Theta \in R^{n*L}$ is a measurement matrix, and we wish to recover the sparse signal $x \in R^L$ such that

$$y = \Theta x.$$

In the earliest studies of compressive sensing, the sparse signal recovery is cast into a linear programming problem:

$$\min_x |x|_1 \quad s.t. \quad |\Theta x - y|_2 \leq \epsilon, \tag{3}$$

where $|.|_1$ is the l_1 penalty function.

The sparse signal recovery could also be relaxed like LASSO, which aims to solve the following unconstrained l_1 regularized minimization problem by a non-linear procedure:

$$\min_x 1/2|\Theta x - y|_2^2 + \lambda|x|_1, \tag{4}$$

where λ is a regularization parameter.

When Θ satisfies the restricted isometry property (RIP)[4], one can reconstruct x by solving the above l_1 minimization problem[3][8]. The standard CS theory indicates that, an $s-$sparse vector in R^L can be recovered efficiently using $m = O(s\log(L/s))$ measurements[4].

3. OUR PROPOSED METHOD-TDCS

For our topic modeling problem, we want to build a topic representation upon the low level semantic concepts, and the training data are the keywords and the unlabeled text data.

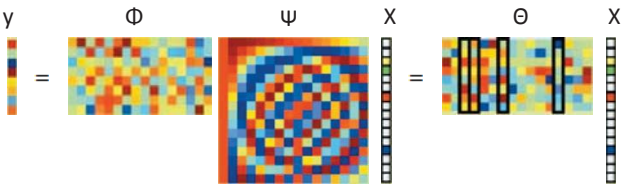

Figure 1: Illustration of Compressive Sensing.

3.1 Problem Definition

Previous topic modeling methods such as PLSA and LDA, model the documents as a mixture of hidden topics/latent semantic concepts, and each topic is usually a probability distribution over the words, w.r.t. $p(w|z)$, where z is the given topic, and w is the words in the vocabulary.

We aim to build a topic representation from several keywords upon the low level semantic concepts with the exploitation of semantic sparsity.

Specifically, given a dataset $D = \{d_1, ..., d_N\}$, and the keywords set $KS = \{keyword_1, ..., keyword_m\}$, we try to learn a topic T_{ks}, which is a distribution over low level semantic concepts/topics z, w.r.t. $p(z|T_{ks})$. T_{ks} is the topic described by the user provided keywords set KS, and z is the lower topics or semantic concepts mined from the documents via a unsupervised approach, e.g. LSI. Under the hypothesis of sparsity, most of the elements of the $p(z|T_{ks})$ are 0. Essentially, we want to refine the semantic concepts for the topic.

3.2 TDCS Framework

The reasonable hypothesis is that the topic is sparse in the semantic space. With the exploitation of the sparsity, we aim to obtain a more concise topic representation. That not only just means a more accurate topic, but also a significant improvement in computation efficiency. The challenging is that the number of keywords is usually very small and random.

We propose a TDCS(Topic Distilling with Compressive Sensing) framework for the problem. In the framework, the keywords are taken as the measurements, and the topic is constructed in the transformed semantic space. One-hot representation is used for each keyword's representation in word space, and they are projected into semantic space to build the measurement matrix. The value of the measurement is the probability or the confidence about the keyword's relevance with the target topic. In this paper, we set it as the default value 1. Of course, the user could provide the probability himself.

Though CS has excellent properties for this kind of task, there is still a theoretical guide for efficient construction. Ideally, based on the sparsity hypothesis, if the provided keywords are accurate and sufficient enough, the topic could be reconstructed efficiently. In practice, there are two crucial issues needed to care about. Firstly, the provided keywords may not be accurate enough. Secondly, the number of provided keywords may be smaller than the theoretical bound, though the bound is not very tight. Both the two issues could result in not reliable results. Therefore, in our problem, an iterative learning approach is adopted to make the result more robust.

The detailed framework and procedure is shown in Figure 2. A unsupervised method e.g. LSI is used to build the original latent semantic space. The keywords are taken as the training data. Firstly, the keywords in word space are projected into semantic space. Compressive sensing is utilized to construct the initial topic representation from them. After the initial topic representation generated, the topic will be reflected into word space in the feedback stage. In feedback stage, we choose the words most likely relevant with the topic to enrich the keywords set, i.e. updating the training data. Then with new keywords set, the learning procedure is conducted once again. The process will be repeated several times. If the stop criteria is satisfied, the iteration will end.

For the convenience to understand the formulation and our proposed method, we list the detailed notation of each variables in Table 1.

Table 1: Notation of the Variables

variable	describtion
D	the dataset
X	document-term representation of the dataset
d	the document
Voc	the vocabulary of the dataset
N	the size of the vocabulary
KS	the set of keywords
m	number of keywords
K	the dimension of LSI's semantic space
L	the dimension of original signal space
s	the dimension of sparse signal
y	corresponding measurement values
A	the measurement matrix
T_{ks}	the topic implied by KS
$p(z\|T_{ks})$	the topic representation of T_{ks}
z	the semantic concepts/topics mined from data

4. TDCS_LSI

In this paper, we take the LSI to build the latent semantic space, so we call the method TDCS_LSI here. Our method could be easily extended into the semantic space built by other methods.

4.1 Latent Semantic Space Building

LSI is used to build the latent semantic space from the document-term matrix X. Ideally, a larger dataset could obtain a better semantic space. The common used term representation method is TFIDF[18], however, for some tasks, there are other better methods.

Through LSI, the term-document matrix X will be decomposed into three components. For matrix V, each dimension v of $V = (v_1, ..., v_K)$ corresponds to a particular semantic concept and they are orthogonal with each other. The dimension v of V could be viewed as the semantic concept/topic z described in Section 3.1.

Since the matrix V is orthogonal, according to CS theory[4], the RIP criteria will be surely satisfied. V will be taken as the semantic basis, which corresponds to matrix Ψ in Figure 1.

The matrix size of X is usually very large and sparse for large data set, so we use an approximated algorithm PSVD[1] for LSI, which pursuits the singular vectors corre-

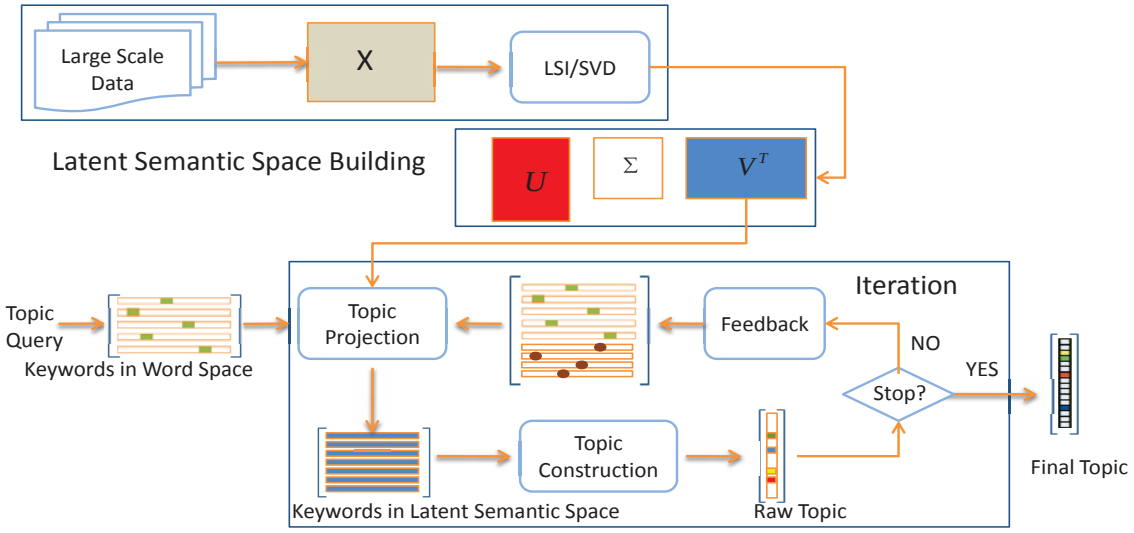

Figure 2: Procedure of the proposed TDCS method.

sponding to the K largest singular values. K is an empirical value and should be carefully set.

4.2 Keywords Projection to Semantic Space

The keywords are taken as the measurements, and in this section, we will explain in detail how to build the measurement matrix through projecting keywords from word space to semantic space.

To represent the keyword of KS in word space, common used one-hot representation method is utilized. Specifically, for $keywrod_i$, the corresponding representation is

$$e_{idx_i} = (0 \ldots, 1, \ldots, 0)^T,$$

where idx_i is the index of the $keyword_i$ in the vocabulary Voc, and the corresponding element is 1 while others are 0.

All the keywords will compose the matrix

$$e = (e_{idx_1}, e_{idx_2}, \ldots, e_{idx_m})^T, \tag{5}$$

where m denotes the number of keywords, and each row corresponds to a keyword.

For $keyword_i$, the projection from word space to semantic space will be

$$a_i = e_{idx_i} V,$$

a_i is the semantic distribution of $keyword_i$. Since each row of matrix V could be viewed as a words' semantic distribution in the latent semantic space, the projection essentially selects the corresponding semantic distribution of $keyword_i$.

The full projection of the keywords e will be

$$A = eV, \tag{6}$$

where each row of A i.e. a_i corresponds to a keyword's semantic distribution.

Under the CS framework, the keywords are taken the measurements, e is the matrix Φ in Figure 1 and the matrix A is exactly the measurement matrix Θ in Figure 1.

The corresponding measurement value for a provided keyword is the user's confidence about the relevance between the keyword and the interesting topic. The measurement value could be written as the vector below

$$y = (y_1, y_2, \ldots, y_m)^T, \tag{7}$$

where $y_i = relevance(keyword_i, T_{ks}) \in (0, 1]$.

Ideally, the user should provide different confidence value to each keyword. In practice, we take 1 as the default value which means all the keywords user provided are strongly relevant with the topic.

The training data KS could be written as a set of pairs, e.g. $\{keyword_i, y_i\}$.

4.3 Topic Construction with CS

The intuition is that the topic is sparse in the latent semantic space. Therefore, we try to find a subspace in V that can best construct and approximate the topic.

The distribution of topic T_{ks} is $p(z|T_{ks})$, for convenience, here we use α instead.

$$\alpha = (\alpha_1, \ldots, \alpha_K)^T,$$

α_i measures how important the corresponding semantic concept v_i contributes to the target topic.

For each pair $\{keyword_i, y_i\}$ in the keywords set KS, the equation below should be satisfied.

$$a_i \alpha \approx y_i \tag{8}$$

Thus, for all pairs $\{keyword_i, y_i\}(i = 1, \ldots, m)$, we need to find a α, such that,

$$A\alpha \approx y, \tag{9}$$

where $m << K$.

This is a standard compressive sensing problem. The matrix A is the measurement matrix, corresponding to Θ, and the y is the measurement value. α corresponds to the x in Figure 1.

Following compressive sensing approach, the optimization problem will be

$$\alpha_* = arg \min_{\alpha} |A\alpha - y|_2^2 + \lambda |\alpha|_1, \tag{10}$$

where y is set as 1 for all keywords in default as we described in last section. λ is the regularization parameter, which controls the degree of the sparsity. A larger λ always means a more sparse result.

The above problem is a sparse optimization problem because of the l_1 term, and in order to solve it efficiently, the package SLEP[1] is facilitated. The details of the package are described in [16] which implements several state-of-art methods for this kind of optimization problem.

λ is a very common parameter for LASSO as well as other sparse models, it's usually tuned in an interval via cross validation in the development set. We will give more detailed description about the parameter tuning in the experiments session.

This approach guarantees that α_* is sparse, i.e., most elements of $\alpha*$ is zero. That's consistent with our previous assumption that only a few latent semantic concepts contribute to a topic.

4.4 Iterative Learning Approach

Though the compressive sensing has excellent properties in many applications. However, in practice, sometimes, the number of keywords is too small to yield reliable results. It's possibly because the provided keywords are smaller than the theoretical bound since the robust reconstruction needs sufficient measurements according to compressive sensing theory. The accuracy of the provided keywords is also an important issue for the topic construction.

Thus, an iterative approach is adopted to make the compressive sensing algorithm more robust for our task. This is a heuristic method to approximate a more stable topic representation.

To be specific, as shown in Figure 2, when an initial topic is generated through compressive sensing, we reflect the topic into word space in the feedback stage. For the reflection, the equation below is utilized,

$$\omega_* = V\alpha_*, \tag{11}$$

the parameter ω_* is the weight of words.

In feedback stage, we choose the words highly relevant with the topic as incremental keywords, w.r.t. IKS. The measurement value y_j of $keyword_j$ in IKS is set as the weight of the $keyword_j$ in ω_* after normalization.

IKS is added into the original keywords set KS to compose a new training set.

$$KS' = KS \cup IKS \tag{12}$$

where $IKS = \{\{keyword_j, y_j\} \dots\}$.

Through several rounds of compressive sensing and feedback, the topic will tend to stable.

During the iteration, an important factor is the size of the IKS, w.r.t. δ.

$$\delta = |IKS|_0 \tag{13}$$

If it's too small, the iteration will be too slow, however, if it's too large, the iteration will shock and lead to the deviation of real topic. The size is fine tuned in experiments by cross validation.

Another important issue is when to stop the iteration. We could calculate the difference value between the last two topic's distribution, if the differ is small than a predefined threshold value, a converged solution is obtained. This method is sensitive to the threshold value which should be carefully tuned. An alternative method is to measure the difference in the word space, if the IKS almost couldn't bring in new keywords for KS, we could consider to stop the iteration. It's usually equal with former one, and more practical. So, we take the latter one for our iterative strategy.

Through our TDCS method, we get a sparse topic, i.e. α_* from the keywords KS, which captures the core semantic concepts and assign suitable values to them. The detailed algorithm of our method is shown in Algorithm 1.

Algorithm 1 The detailed TDCS algorithm

```
input:
  document-term matrix X
  initial keyword list KS
output:
  topic representation: α*
  w of vocabulary
% latent semantic space building
D is decomposed into UΣV^T
repeat
    for each keyword in KS do
        generate the corresponding e_i
    end for
    generate the matrix e
    % topic projection
    A = e * V;
    % topic α
    find the α* s.t. min  |Aα − y|_2^2 + λ|α|_1
    % topic feedback
    w = V * α*;
    choose the words with highest weight
    form an incremental keyword list IKS
    KS' = KS ∪ IKS;
    % update the training data
    KS = KS'
until IKS − KS ≈ NULL
```

4.5 TDCS for Text Mining Tasks

With our TDCS method, the topic $p(z|T_{ks})$ i.e. α_* is learnt from the keywords KS, which could be used for many subsequent text mining tasks with the benefits both in accuracy and efficiency, for example, text classification, information retrieval.

4.5.1 Text Mining Tasks

Text Classification

For text classification, the interesting topic will be the category(class) in a dataset. It becomes the problem called text classification by labeling words [15][11], existing methods are all tackling in the word space.

Usually, a linear classification model could be written as $g(f_w(x_i))$, where

$$f_w(x_i) = \omega^T x_i, \tag{14}$$

and $g(.)$ could be 0/1 function, sigmoid function, etc.

The loss function of for training the linear classification could be

$$J(w) = \sum_i L(y_i f_w(x_i)) + \lambda R(w)$$

[1]http://yelab.net/software/SLEP/

where the L could be square loss, log loss, hinge loss, etc. The $R(w)$ is the regularization term.

$f_w(x_i)$ is the most important part for these kinds of classifiers, which is the product of the parameter w and x_i, measuring the similarly between an instance and target class. For text classification, usually, x_i is the BOW representation of a document d_i, the classifiers try to learn a optimal w_* from the training data for $f_w(x_i)$.

Here, the training data is the keywords, different with existing methods, our method could skillfully transform the problem into semantic space. For one class classification, by projecting documents x into a K-dimensional semantic space, w.r.t. x', the w' will be learnt in the K-dimensional semantic space. The α_* learnt by our method is a very good choice as the w'_* for $f_{w'}(.)$.

Information Retrieval

For information retrieval, the keywords KS are treated as the query, w.r.t. q, the VSM[23] model will represent the query q and document d in word space, and then calculate the similarity between them with

$$sim(q, d) = q \cdot d. \tag{15}$$

For LSI, both the query and document in word space are projected into semantic space w.r.t. q' and d' following Equation 2, and then $sim(q', d')$ is calculated similar with VSM by Equation 15.

With our method, the query q is refined in the semantic space, a more accurate and concise q'' i.e. α_* could be used for the similarity calculation as below.

$$sim(q, d) = \alpha_* \cdot d' \tag{16}$$

4.5.2 Computational Efficiency

The main computation for text classification or information retrieval is the product between Q and $R(d)$, w.r.t. $Q \cdot R(d)$, here Q stands for w in $f_w(.)$ or q in $sim(q, d)$, $R(d)$ stands for x_i in $f_w(.)$ or d in $sim(q, d)$.

The original dimension of Q and $R(d)$ is N, N is the size of vocabulary. After the projection, N will be reduced to K. With our method, the effective dimension of Q for product calculation will be s, $s < K$. Usually, s is one order smaller than K. That means a great gain in computational complexity.

With our method, the theoretical time complexity for production will decrease from $O(N * M)$ or $O(K * M)$ to $O(s * M)$, where M is the size of dataset.

5. EXPERIMENTS

Since the topic representation is hard to evaluate directly, instead, we evaluate it in an indirect way. It will be utilized for concrete text mining tasks, here we choose one class text classification as the task which is similar with information retrieval, and we will verify the effectiveness of our method through the performance of classification.

Since our task here is one class text classification, aiming to identity target category documents from unlabeled set, therefore it's appropriate to use AUC(Area under the Curve of ROC) to measure the performance[9]. A larger AUC indicates better performance.

The keywords generation process will be given in the experiments setting section. We will learn from each keywords set respectively.

All the experiments are performed on a PC with two Intel Xeon E5-2620 @ 2.0GHz CPUs and 8GB memory.

5.1 Datasets

We use the following datasets in our experiments, i.e. **20 Newsgroups**[2], **WebKB**[3] and **Movie Reviews**[4].

- **20 Newsgroups** The 20 Newsgroups data set is a collection of approximately 20,000 newsgroup documents. The data is organized into 20 different newsgroups, each corresponding to a different topic. Some of the newsgroups are very closely related to each other, while others are highly unrelated.

- **WebKB** The WebKB dataset contains web pages gathered from university computer science departments. The web pages are composed of 4 categories: *student*, *course*, *faculty* or *project*.

- **Movie Reviews** This dataset is a collection of movie reviews from IMDB. The movies are labeled with respect to their overall sentiment polarity (positive or negative)[22].

All datasets were processed using lowercased stemmed unigram words, with HTML tags, stop-words and words with length less than three removed.

Table 2 summarizes the datasets statistics after these processing. For each dataset, we randomly select 2/10 of the dataset as the testing data, and use the rest of the dataset as training set.

Table 2: Dataset

data set	instances	categories	words
20 Newsgroups	18828	20	19789
WebKB	4199	4	7770
Movie Reviews	2000	2	13843

5.2 Methods for Comparison

The task here is one class text classification, and the labeled training data is keywords. Existing standard methods for one class classification, such as one-class SVM[17] and PU learning[12] require sufficient labeled documents, so they are not appropriate for the task here.

GE/SWIRL

Since this task is to learn a classifier from keywords, we take GE/SWIRL[11] as the method for comparison, which is a state-of-the-art method for learning from features(words). GE/SWIRL is developed from GE and considers the sequence of the keywords. In this paper, the sequence is not considered. The original method of GE/SWIRL is designed for multi-class classification with multiple keywords sets, for convince, we modify it for one class classification by set the target class value of each keyword as 0.99 while other classes as 0.01.

VSM and LSI

If we take the documents which belong to the target category as the relevant documents for the user provided

[2]http://people.cs.umass.edu/m̃ccallum/data.html
[3]http://www.cs.cmu.edu/w̃ebkb
[4]http://www.cs.cornell.edu/people/pabo/movie-review-data/

keywords. This task could be viewed as a coarse-granularly information retrieval. Therefore, we also take the VSM and LSI as the methods for comparison.

For our TDCS method, the classifier is the $g(f_w(x))$, where $g(.)$ is 0/1 function, and $f_w(x) = \alpha_* \cdot x$.

For all the methods, one keywords set is trained each time as well as our TDCS method.

5.3 Experiment Settings

5.3.1 Topic Description(Keywords) Generation

user provided approach

Some real user provided keyword lists[11] are utilized for **WebKB** and **Movie Reviews**. In [11], the labeled keywords are formatted as list of {keyword,label} pairs. The labels are the categories. We extract the keywords from the list as the user generated keywords.

oracle approach

Ideally, a keyword should be highly predictive of the topic. In practice, we assume there exists an oracle that can reveal the keywords. We then select keyword according to their predictive power measured by TFIDF score of the given category. This is applied for **20 Newsgroups** because there're no public human labeled keywords for this dataset.

We created a keywords pool for each topic(category) of the datasets, the keywords are collected with above approaches. During the experiments, the keywords for a topic are randomly selected from the pool.

5.3.2 Implementation Details

All documents are represented by the vector of words. For term weighting method, **20 Newsgroups** and **WebKB** is tfidf , and **Movie Reviews** is 0/1, i.e. the occurrence of a word or not. The reason why we use 0/1 representation for **Movie Reviews** is that it's designed for sentiment classification, and there have been some works shown that for sentiment classification, the 0/1 representation is better than tfidf upon most of classification models[14].

Once the document-term matrices of training data generated, LSI is applied to build the semantic space. The dimension of the semantic space, i.e. K is empirically set as 200, which is a commonly used value[18].

The parameter λ in Equation 10 and δ in Equation 13 are fine tuned via cross validation. We randomly select 1/10 of the training data as the development set and conduct the parameter tuning in the development set. The parameter λ is tuned firstly without iteration, and then the δ is tuned.

The tuning procedure is simple, several keywords sets are chosen to build the models with different parameter values. The models will be utilized in the development dataset, after the evaluation of their performance, the parameter value of the best model is set as the final value.

The parameter λ is tuned in $[1e-5, 1e-4, 1e-3, 1e-2, 1e-1, 1]$, and δ is tuned in the interval $[1, \frac{m}{2}]$. After tuning, the value of λ used in the experiments is 0.01, and the value of δ is $\lfloor \frac{m}{2} \rfloor$.

5.4 Experiment Results

We evaluate the performance with different topic descriptions as well as different number of keywords, and give detailed analysis of the iterative approach.

For each category of the datasets, we repeated the experiments 100 times. In order to investigate general influence by the number of keywords, we randomly choose 5 and 10 keywords respectively each time to describe the topic in Movie Reviews. For 20 Newsgroups, we randomly choose 5 keywords as the topic description each time. For WebKB, we randomly choose keywords with variable-length from 5 to 10 each time. Table 3 summarizes the keyword numbers for each dataset.

The average results in three benchmark datasets is shown in Table 4[5]. The corresponding keywords setup is shown in Table 3. The only difference between Movie Reviews(k1) and Movie Reviews(k2) is the number of keywords provided.

Table 3: number of keywords for each dataset

dataset	number of keywords
Movie Reviews (k1)	5
Movie Reviews (k2)	10
20 Newsgroups	5
WebKB	5-10

Table 4: average results of three datasets

dataset	VSM	LSI	GE/SWIRL	TDCS
Movie Rev. (k1)	0.528	**0.545**	0.540	**0.596***
Movie Rev. (k2)	0.561	0.572	**0.584**	**0.619***
20 Newsgroups	0.578	**0.918**	0.891	**0.944***
WebKB	0.705	0.784	**0.802**	**0.825***

5.4.1 General Results

We can observe that in all the datasets and with different keyword setups, our TDCS method outperforms the other methods[6], which demonstrates the effectiveness and robustness of our method, and reveals that the distilled topic is more precise for text classification.

We can also find that when number of keywords is small, e.g. 5 in **Movie Reviews** and **20 Newsgroups**, the LSI is better than GE/SWIRL. However, GE/SWIRL will be better when the number increases, e.g. 10 in **Movie Reviews** and 5-10 in **WebKB**. A possible reason is that the LSI is superior in improving the recall especially when the number of keywords is too small. However, when the number of keywords increase, the impact of this aspect will be not obvious.

Essentially, in comparison with LSI, our method not only takes advantage of the semantic space to improve the recall, but also utilizes the semantic sparsity to distill important semantic components from the semantic space which lead to the improvement of precision.

The general results in **Movie Reviews** are not very good compared with the other two datasets, a possible reason is that the latent semantic space built by LSI is not ideal for sentiment classification. It's hard to guess what the basis vectors in the LSI space may correspond to. However, from the experiment results, it seems that there should be some semantic concepts highly relevant with sentiments since our

[5]The best and second best results are bold in text.

[6]* The improvement is statistically significant, since the p value < 0.05 according to t test.

method could get better results than others. With a better semantic space for sentiment analysis, the results of our method will be surely better.

In **20 Newsgroups**, we can find that LSI, GE/SWIRL and our TDCS method all get good results compared with the other datasets. The main reason is the generation approach of the keywords. We choose the keywords with high tfidf values as the topic descriptions, they are representative for the category and discriminative with other categories.

5.4.2 General Quantitative Analysis

From the results of **Movie Reviews** in Table 4, we can find that, roughly, when the number of keywords grows, the performance grows. The average result with 10 keywords is better than that with only 5 keywords. This suggests that for a robust topic construction with CS, sufficient measurements are needed.

Less Wins More: From Table 4 we can also find that our method can get better results with 5 keywords than LSI and GE/SWIRL with 10 keywords. This shows the advantage of our method especially when training data is small. With less training data, here is the 1/2 of the training data, our method could still get comparable and even better results. This very useful, since people are used to provide as few keywords as possible for a topic.

5.4.3 Detailed Results on Movie Reviews

In Table 5, we illustrate some detailed examples from the results in movie data. The detailed topic descriptions are shown in Table 6. These descriptions are all for the 'positive' category.

Table 5: results of examples on movie reviews

index	VSM	LSI	GE/SWIRL	TDCS
T_1	0.508	0.502	**0.510**	**0.627**
T_2	**0.514**	0.506	0.510	**0.518**
T_3	0.580	**0.671**	0.647	**0.803**
T_4	0.611	0.626	**0.642**	**0.743**
T_5	0.551	0.564	**0.610**	**0.664**
T_6	0.550	0.568	**0.594**	**0.647**

Table 6: keyword examples of the movie reviews

index	topic desc.
T_1	talent, cool, alwai, laugh, great
T_2	heartfelt, high, esteem, worth, provoc
T_3	oscar, uniqu, easili, touch, esteem
T_4	worth, inspir, entertain, great, amaz, uniqu, emot, success, astound, imagin
T_5	love, modern, interest, nice, award, cclaim, top, success, excruciatingli, captiv
T_6	heartwarm, must, incred, adventur, touch, special, represent, excel, provoc, funni

We can observe that the results are sensitive to the keywords provided, even though the number of keywords are the same with each other. For example, the keyword 'high' in T_2 is not very good to indicate the positive sentiment, so the result is worse than others. Though T_3 only has 5 keywords, it could get the better result than those with 10 keywords. This reveals that the quality of the keywords is also an important factor for our method. In other words, the provided keywords should be representative for the topic.

The performance of topic T_1 (5 keywords) with our method, is better than T_4, T_5 and T_6 (10 keywords) with SVM or LSI method. This also demonstrates the effectiveness of our method, with 1/2 of training data, our method can get better results than those with full data.

5.4.4 Detailed Results on WebKB

Table 7 shows some detailed results on WebKB for 'faculty' category.

The keywords are given with variable length, $T_4 > T_3 > T_2 = T_1$. However, the result is not strictly in proportion to the number of keywords. That's reasonable, take example for T_1 and T_2, they are better than T_3. The word 'depart' and 'email' in T_3 are also possible in the 'student' category. This also suggests that good quality of the keywords is essential.

Table 7: keyword examples of the WebKB data

index	topic desc.	TDCS
T_1	teacher, staff, professor, ta, teach	0.843
T_2	advisori, editori, profession, director, coauthor	0.851
T_3	teach, experi, educ, depart, email, graduat	0.712
T_4	teacher, professor, career, job, faculti, depart, staff	0.860

5.4.5 Impact of Iteration on 20 Newsgroups

We give an example in 20 Newsgroups to show the impact of our iterative learning approach. The topic description T_1 and T_2 in Table 8 are both generated for the 'auto' category. Figure 3 shows the performance of text classification after each iteration. In general, the performance increases along with the increase of iteration numbers.

Table 8: keyword examples of the 20 Newsgroups

index	topic description
T_1	gear, bumper, chrysler, mazda, overpass
T_2	sedan, clutch, jetta, ferrari, dealer

Keywords Expansion: We also list the keywords which are mostly relevant with the topic description T_1 and T_2 respectively in Table 9. The keywords are picked from the final expanded set KS, and for each topic, we choose the top 20 keywords. We can find that if the user really have an interest on this topic, and give accurate descriptions, though the initial given keywords are different, the final expanded relevant keywords are very close. This gives an insight for better keyword recommendation.

Ideally, for a topic, if the descriptions are accurate enough, after the iteration, the constructed topic representations in semantic space will tend to be close. In practice, due to the semantic diversity of the keywords provided by different users, the difference between the distilled topic representations is unavoidable. The iterative learning approach could reduce the difference.

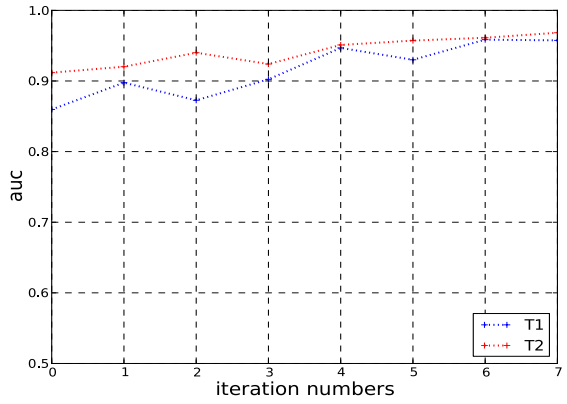

Figure 3: Classification performance along iteration

Table 9: Expanded Keywords on 20 Newsgroups

topic desc.	T_1	T_2
relevant keywords	car, opel, detector, owner, version, price, clutch, engin, color, nissan, auto, dealer, manual, ford, widget, model, drive, oil, toyota, speed	car, version, detector, engin, clutch, nissan, drive, auto, opel, ford, dealer, oil, honda, model, mile, bmw, toyota, speed, insur, audi

5.4.6 *Computation Cost for Text Classification*

Table 10 shows the average classification time for each dataset. The time duration is measured in seconds. We can see that with our method, the computation cost is smaller in a order of magnitude.

The computation is conducted with Matlab, which optimizes the matrix computation in memory, and datasets used in our experiments are tiny in comparison with real world datasets, therefore the total computation time is little. However, for real web data, which contains millions of documents, the difference will absolutely become larger, e.g. 1 hour vs. 10 hour.

Table 10: Classification Time

Data Set	VSM	LSI	TDCS
20 Newsgroups	7.45e-2	8.96e-4	1.29e-4
WebKB	9.6e-3	2.30e-4	5.69e-5
Movie Reviews	9.0e-3	1.78e-4	4.61e-5

Another issue for big data is that it's impossible to load all the data once a time. The frequent data throughput is essential, which is usually time-consuming. With the effective dimension reduction through our method, more data could be loaded into the memory each time. This is very efficient for text classification, information retrieval, etc.

Summary: We have presented comprehensive experiments using three different text collections to evaluate the effectiveness of the proposed TDCS method. In general, with more keywords, the result will be more accurate. The quality of the keywords is also an important issue. Moreover, with the reduction in the topic representation space, the computation complexity significantly reduces.

6. RELATED WORK

Our work focuses on the topic construction in the latent semantic space, in this paper, we take LSI to build the semantic space, and surely our method could be extended to other semantic spaces. Meanwhile, there have been arising works on the utilization of semantic sparsity in text mining recently.

6.1 Semantic Representation

There have been other methods for construct latent semantic concepts/topics, for example, PLSA[10] and LDA[2]. They are the typical techniques aiming to find useful latent semantic structure in the unstructured collection. They assume that each document exhibits multiple topics, yet ignores the correlation between topics. PLSA could be formulated a generation process including the document topic distribution and topic word distribution. It's highly sensitive to task domain, which is continuously changing in real documents. Different with PLSA, LDA uncovers the underlying semantic structure based on a hierarchical Bayesian analysis of the original texts. Thus, it becomes computationally very expensive on large data sets.

Word2Vec has emerged as a new technique to learn the distributed representation of words. The neural language model is used to represent the documents, and the learning algorithm is a combination of neural networks and softmax model[19]. It's a popular word embedding method, and was previously better than LSI for preserving linear regularities among words. [21] tells us that word2vec is essentially equivalent with the implicit matrix decomposition with some mathematical explanations.

These works give us a probability to extend our method into semantic representation space built by other methods.

6.2 Semantic Sparsity

In [5], a sparse latent semantic analysis model is proposed for text analysis, in which, the keyword distribution of each topic is sparse. In [20], they decompose the topic into low-rank principle component and sparse document component. The work is conducted on the assumption that each document has its own specific keywords which is sparse in comparison with other documents. In [25], a regularized LSI was proposed to obtain sparse semantic concepts(topics) belong to a document, aiming to scale up LSI for large document sets. In [24], a variation model of PLSI and LDA is proposed to fully leverage the sparsity of topics belonging to a document, which is efficient for large scale data.[13] propose a dual-sparse topic model with the hypothesis that in real-world scenarios, individual documents usually concentrate on several salient topics instead of covering a wide variety of topics. A real topic also adopts a narrow range of terms instead of a wide coverage of the vocabulary.

All these works are tackling the sparsity on documents or topics or words, and building models from the document data set. Our work is different with these works. Firstly, our mission is to construct a more concise topic upon the latent semantic space with the sparsity assumption. Secondly, we need to build the topic from only several keywords rather than documents.

7. CONCLUSION

In this paper, we propose a method called TDCS to distill the topic from dynamically provided keywords. The sparsity of the topic is exploited and utilized, and CS is skillfully applied to establish a sparse structure in the semantic space for the topic. With our method, a more precise topic could be obtained. Besides, our method is efficient for subsequent tasks with the significant reduction of computation.

The experiments on several benchmark datasets demonstrate the effectiveness of our method. Besides text classification, the TDCS method could be used for many other applications, for example, information retrieval, query understanding, keywords recommendation, since which could provide more accurate semantic information.

Our approach and results give promising future research direction towards the investigation of sparsity in text analysis. We would like to extend our work using other semantic representation methods in the future, and further explore the theoretical perspectives about this method.

8. ACKNOWLEDGMENTS

This work is supported by the National Science Foundation of China(No.61321491, No.61202320), National Social Science Foundation of China(No.11AZD121), and Natural Science Foundation of Jiangsu Province(No.BK2012304). It was also partially supported by Collaborative Innovation Center of Novel Software Technology and Industrialization.

9. REFERENCES

[1] M. W. Berry. Large scale sparse singular value computations. *International Journal of Supercomputer Applications*, 6:13–49, 1992.

[2] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3:993–1022, Mar. 2003.

[3] E. J. Candes, J. Romberg, and T. Tao. Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information. *IEEE Transactions on Information Theory*, 52(2):489–509, Feb. 2006.

[4] E. J. Candes, J. K. Romberg, and T. Tao. Stable signal recovery from incomplete and inaccurate measurements. *Communications on pure and applied mathematics*, 59(8):1207–1223, 2006.

[5] X. Chen, Y. Qi, B. Bai, Q. Lin, and J. G. Carbonell. Sparse latent semantic analysis. In *Proceedings of the 2011 SIAM International Conference on Data Mining*, pages 474–485, 2011.

[6] S. C. Deerwester, S. T. Dumais, T. K. Landauer, G. W. Furnas, and R. A. Harshman. Indexing by latent semantic analysis. *JASIS*, 41(6):391–407, 1990.

[7] D. L. Donoho. Compressed sensing. *IEEE Transactions on Information Theory*, 52:1289–1306, 2006.

[8] D. L. Donoho. For most large underdetermined systems of linear equations the minimal l1-norm solution is also the sparsest solution. *Communications on Pure and Applied Mathematics*, 59(6):797–829, 2006.

[9] J. A. Hanley and B. J. McNeil. The meaning and use of the area under a receiver operating characteristic (roc) curve. *Radiology*, 143(1):29–36, 1982.

[10] T. Hofmann. Probabilistic latent semantic indexing. In *Proceedings of the 22Nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '99, pages 50–57, New York, NY, USA, 1999. ACM.

[11] K.-S. Jun, J. Zhu, B. Settles, and T. Rogers. Learning from human-generated lists. In *Proceedings of the 30th International Conference on Machine Learning*, ICML'13, pages 181–189, 2013.

[12] X. Li and B. Liu. Learning to classify texts using positive and unlabeled data. In *IJCAI*, volume 3, pages 587–592, 2003.

[13] T. Lin, W. Tian, Q. Mei, and H. Cheng. The dual-sparse topic model: mining focused topics and focused terms in short text. In *Proceedings of the 23rd international conference on World wide web*, pages 539–550. International World Wide Web Conferences Steering Committee, 2014.

[14] B. Liu. Sentiment analysis and opinion mining. *Synthesis Lectures on Human Language Technologies*, 5(1):1–167, 2012.

[15] B. Liu, X. Li, W. S. Lee, and P. S. Yu. Text classification by labeling words. In *AAAI*, volume 4, pages 425–430, 2004.

[16] J. Liu, S. Ji, and J. Ye. *SLEP: Sparse Learning with Efficient Projections*. Arizona State University, 2009.

[17] L. M. Manevitz and M. Yousef. One-class svms for document classification. *the Journal of machine Learning research*, 2:139–154, 2002.

[18] C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, USA, 2008.

[19] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[20] K. Min, Z. Zhang, J. Wright, and Y. Ma. Decomposing background topics from keywords by principal component pursuit. In *Proceedings of the 19th ACM International Conference on Information and Knowledge Management*, CIKM '10, pages 269–278, New York, NY, USA, 2010. ACM.

[21] Y. G. Omer Levy. Neural word embedding as implicit matrix factorization. In *Advances in Neural Information Processing Systems 27*. Curran Associates, Inc., 2014.

[22] B. Pang and L. Lee. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of the ACL*, 2004.

[23] G. Salton, A. Wong, and C. S. Yang. A vector space model for automatic indexing. *Commun. ACM*, 18(11):613–620, Nov. 1975.

[24] K. Than and T. B. Ho. Fully sparse topic models. In *Machine Learning and Knowledge Discovery in Databases*, pages 490–505. Springer, 2012.

[25] Q. Wang, J. Xu, H. Li, and N. Craswell. Regularized latent semantic indexing. In *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '11, pages 685–694, New York, NY, USA, 2011. ACM.