Large Scale of E-learning Resources Clustering with Parallel Affinity Propagation

Wenhua Wang, Hanwang Zhang, Fei Wu, and Yueting Zhuang

College of Computer Science and Engineering, Zhejiang University, Hangzhou, P.R.China
turningice@gmail.com,ashmate123@yahoo.com.cn,wufei@cs.zju.edu.cn,
yzhuang@cs.zju.edu.cn

Abstract. E-learning resources increase vastly with the pervasion of the Internet. Thus, the retrieval of e-learning resources becomes more important. However, the typical lexical matching could not satisfy the users' underlying intention. We adapted affinity propagation in MapReduce framework to make semantic retrieval applicable to large-scale data, and with this parallel affinity propagation we proposed an approach to retrieve e-learning materials efficiently, which could retrieve semantically relevant materials utilizing conceptual topics produced in advance.

Keywords: E-learning Resources Retrieval; Parallel Affinity Propagation; MapReduce

1 Introduction and Related Work

With the pervasion of the Internet, e-learning is more and more popular, which provides a brand new way for people to learn without attending face-to-face class. With e-learning, the student and the teacher use online technology to interact, which profits from a combination of techniques including computer networks, multimedia, content portals, digital libraries, search engines, etc. The worldwide e-learning industry is estimated to be worth over 38 billion euros according to conservative estimates [1]. With the prevalence of e-learning, the amount of learning resources also grows exponentially, which makes it not feasible to access them only by clicking links. Thereby, an effective mechanism is needed to locate the resources, with which people could find the e-learning materials they want with facility. To accurately locate the e-learning materials a user is seeking for, a system has to guess the user's underlying intention from the text typed in, rather than merely return the results from literally matching, particularly when the user is not familiar with the terminologies of the field which he/she is trying to learning. Thus, leveraging the data mining technology to locate the resources semantically related to the querying text becomes meaningful. Nowadays, the e-learning resources comprise texts, images, videos, audios and materials in other modalities, however, text materials are the best choice to

Joseph Fong, Reggie Kwan, Fu Lee Wang (Eds.): Hybrid Learning: A New Frontier, pp. 1-10.
ICHL 2008, City University of Hong Kong, Hong Kong, August 2008.

be analyzed and understood, in that texts account for the biggest part and only the text resources reflect the information most directly. Besides, taking efficiency and the expensive mining cost into account, it's reasonable to focus only on the text materials and to neglect materials in other modalities.

In data mining technology, clustering is an effective method to discover clues when little is known about the data. Besides, e-learning materials are intrinsically appropriate to be clustered, in that fields of materials concerning likely overlap fields of others, and materials concerning similar fields probably use the same words, particularly the same terminologies. For example, two physics books will likely use words such as '*energy'*, '*force'*, '*mass'*, and '*charge'* repeatedly, which consequently strengthens the correlation between the books . So we adopt the clustering method to discovery the correlations among the E-learning materials.

Traditionally, measures of text similarity have been used for a long time in applications in natural language processing and related areas [5] .One of the earliest applications of text similarity is perhaps the vector model in information retrieval, where the document most relevant to an input query is determined by ranking documents in a collection in reversed order of their similarity to the given query [6]. In the vector space model, a document is represented by a vector indexed by the terms of the corpus, so two documents that use semantically related but distinct words will therefore show no similarity. Many methods were proposed to explicitly or implicitly make use of similarity between terms, such as [4] [5] [7], and some other WordNet [8] based methods.

Those information retrieval technologies work well with toy data. However, because of memory limitation of stand-alone computer and the expensive computation cost of matrix operation such as singular value decomposition, situation becomes intractable when they are applied to large-scale data, taking unendurable long time or even being interrupted due to out of memory. To tackle the unavoidable problem, usually two categories of methods are used. The first kind is to use matrix factorization and to merge the results using mathematic method, such as [11], [12]. The other kind is to parallelize the learning procedure and to compute each part on distributed computers in parallel. In this paper, we take advantage of MapReduce framework [9], which helps to run specially designed program on distributed computers. After the clustering to the original large data set, we then construct semantic spaces on the resultant relatively small-scale data sets to carry out semantically retrieval on e-learning materials.

In this paper, we proposed a method to manage the e-learning resources and to retrieve the semantically related materials according to users' underlying intention. To tackle the problem induced by the large scale of data, we devise a parallel clustering method in MapReduce framework to preprocess the e-learning materials.

2. E-learning Resources Retrieval

Typically, information is retrieved by literally matching terms in documents. However, this kind of methods can be inaccurate to match a user's query. Since a given concept can be expressed by in many different ways, the literal terms in the

query may not match those of a relevant document, particularly when the user is not familiar with the terminologies of the field he/she is learning about. In addition, most words have multiple meanings, so lexical matching may mistake an irrelevant document as relevant as long as the document contains the same term. A better approach would allow users to retrieve information on the basis of a conceptual topic or meaning of a document [2].

2.1 Conceptual Topic Clustering

As we mentioned in the introduction section, e-learning materials are intrinsically appropriate and straightforward to be clustered into conceptual topics. It's reasonable to think that e-learning materials belonging to different topics have little resemblance. For instance, if the query is "Algorithms and Data Structure", obviously, the results belong in the topic of computer science and materials belonging in other topics such as physics needn't to be searched. In addition, varying from literally matching, methods employing semantic information are time-consuming and use a lot of memory. Therefore, e-learning resources are required to be preprocessed and partitioned into topics before semantic retrieval.

In this paper, we adopt a method called affinity propagation to cluster the e-learning resources. Affinity propagation takes as input a collection of real-valued similarities between data points, and outputs the clustered data by identifying a representative example called exemplar for each data point. Instead of using the original affinity propagation directly, we adapted it in MapReduce framework to make it applicable to large-scale data. The adapted parallel affinity propagation is elaborated in section 3.2.

2.2 Semantic Retrieval in Topic

After e-learning resources are clustered, each cluster has an exemplar which can represent the topic of resources in this cluster. Hence, we construct a two-layer retrieval model using latent semantic indexing (LSI).

LSI is used to overcome the problems of lexical matching by using statistically derived conceptual indices instead of individual words for retrieval [2]. In LSI, truncated singular value decomposition is used to estimate the structure in word usage across documents. To use LSI, a term by document matrix $\mathbf{A}_{t\times d}$ is constructed first, where the value a_{ij} reflects frequency of term i in document j. The matrix $\mathbf{A}_{t\times d}$ is factored into the product of three matrices using the SVD. The SVD of $\mathbf{A}_{t\times d}$, denoted by $SVD(\mathbf{A}_{t\times d})$, is defined as

$$\mathbf{A}_{t\times d} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T \quad (1)$$

The SVD derives the latent semantic structure model from the orthogonal matrices $\mathbf{U}$ and $\mathbf{V}$ containing left and right singular vectors of $\mathbf{A}_{t\times d}$, respectively, and the diagonal matrix $\mathbf{\Sigma}$. These matrices reflect a breakdown of the original relationships into linearly independent vectors or factor values. The use of k factors or k-largest singular triplets is equivalent to approximating the original term-document matrix. In

some sense, the SVD can be viewed as a technique for deriving a set of uncorrelated indexing variables or factors, whereby each term and document is represented by a vector in k-space using elements of the left or right singular vectors. LSI represents terms and documents in the same semantic space, and refer to reference [2], [3], [4] for details. Most of LSI processing time is spent in computing the truncated SVD of the large term by document matrices.

The retrieval process is as follows. When a query is submitted, it is analyzed implicitly and the most relevant topics are returned, which is done by construct a semantic space using the exemplars, since the number of clusters is much smaller than that of e-learning materials, thus, the dimension of the exemplar semantic space is much lower.

For each cluster, topic semantic space is built, and the semantic retrieval is processed with LSI within the several most relevant topics which are returned in the topic retrieval step. Thus, we don't have to do the expensive singular value decomposition step with all the data at one time, and the cost of semantic retrieval is alleviated accordingly.

3. Parallel Affinity Propagation

In this section, we propose a clustering method called parallel affinity propagation, implemented in MapReduce framework. We first introduce the MapReduce programming model, and then apply the programming model to parallelize the standard original affinity propagation. With the proposed parallel affinity propagation, it is feasible to cluster the vast amount of e-learning resources.

3.1 MapReduce Framework

MapReduce is a programming model and an associated implementation for processing and generating large data sets [9]. Users specify a map function that processes a key/value pair to generate a set of intermediate key/value pairs, and a reduce function that merges all intermediate values associated with the same intermediate key. Programs written in this functional style are automatically parallelized and executed on a large cluster of commodity machines. The run-time system takes care of the details of partitioning the input data, scheduling the program's execution across a set of machines, handling machine failures, and managing the required inter-machine communication [9].

The computation takes a set of input key/value pairs, and produces a set of output key/value pairs. The user of the MapReduce library expresses the computation as two functions: Map and Reduce. Map, written by the user, takes an input pair and produces a set of intermediate key/value pairs. The MapReduce library groups together all intermediate values associated with the same intermediate key I and passes them to the Reduce function. The Reduce function, also written by the user, accepts an intermediate key I and a list of values for that key, which makes sure that values with same key are processed at one time on the same computer. Reduce function merges together these values to form a possibly smaller set of values. The

intermediate values are supplied to the user's reduce function via an iterator. This allows us to handle lists of values that are too large to fit in memory [9].

3.2 Adaptation of Affinity Propagation

Affinity propagation is a clustering method, which starts by considering all the data points as potential exemplars, and then recursively transmits real-valued messages along edges of the network whose nodes are data points. At any point in time, the magnitude of each message reflects the current affinity that one data point has for choosing another data point as its exemplar [10]. After certain number of iterations, a good set of exemplars and corresponding clusters emerges. The input of affinity propagation is a collection of real-valued similarities between data points, where the similarity $s(i, k)$ indicates how well data point k is suited to be the exemplar for point i. In affinity propagation, the number of clusters is not required to be specified, which could be influenced implicitly by adjusting values of $s(i, i)$ called "preference". The data point with larger value of $s(i, i)$ is more likely to be chosen as an exemplar. There are two kinds of messages are exchanged between data points, namely "responsibility" and "availability". The "responsibility" $r(i, k)$, sent from point i to point k, reflects how well-suited data point k is to serve as the exemplar for data point i. The "availability" $a(i, k)$, sent from candidate exemplar point k to point i, reflects the accumulated evidence for how appropriate it would be for point i to choose point k as its exemplar, taking into account the support from other points that point k should be an exemplar [10]. The updating is according to the following rules:

$$r(i,k)=(1-\lambda)\times r(i,k)+\lambda\times\left(s(i,k)-\max_{k's.t.k'\neq k}\left\{a(i,k')+s(i,k')\right\}\right) \tag{2}$$

$$a(i,k)=(1-\lambda)\times a(i,k)+\lambda\times\left(\min\left\{0,r(k,k)+\sum_{i's.t.i'\notin\{i,k\}}\max\{0,r(i',k)\}\right\}\right) \tag{3}$$

$$a(k,k)=(1-\lambda)\times a(k,k)+\lambda\times\sum_{i's.t.i'\neq k}\max\{0,r(i',k)\} \tag{4}$$

In the three equations above, λ is the damping factor used to avoid numerical oscillations. To begin with, the availabilities are initialized to zero: $a(i, k)=0$. The detailed explanations to these equations could be found in reference [10]. In this paper, we only focus on how to parallelize the updating process now that the computation of "responsibility" and "availability" depend on each other recursively and tightly. If we represent the "responsibility" and "availability" in the form of matrix, we can find by analyzing three equations above that the value of $r(i, k)$ depends on the entire row of similarity matrix and "availability" matrix, namely $s(i, :)$ and $a(i, :)$. Similarly, the value of $a(i, k)$ depends on the entire column of "responsibility" matrix, namely $r(:, k)$. Therefore, it's not possible to split data points into several partitions and compute their "responsibility" and "availability" values respectively, which corresponds well to the fact that the computation of "availability" of each data point need to collect the support from all other points. The same situation is also applicable to "responsibility".

There are some constraints to be considered when parallelizing affinity propagation:

1. The entire row of "responsibility" should be calculated on the same computer, in respect that each value of $r(i, k)$ takes the same entire *i-th* row of similarity and "availability" as input.
2. Similarly, the entire column of "responsibility" should be calculated on the same computer, since each value of $a(i, k)$ depends on the entire *k-th* column of "responsibility".
3. Considering every value should to be damped, it is necessary to make sure the corresponding value calculated in the last iteration should be kept for the current iteration.

Since it's not possible to parallelize the global computation by splitting input data into pieces and calculating each piece respectively, we intend to parallelize computation within each iteration.

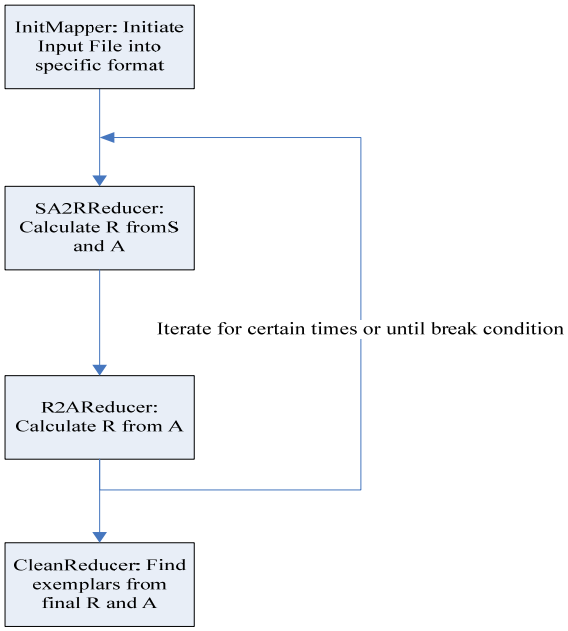

Fig.1. Flow Chart of Parallel Affinity Propagation

Figure 1 shows the process of parallel affinity propagation. Each box represents a step respectively corresponds to step 1 to step 4. Each step mainly comprises a mapper class and a reducer class in Hadoop implementation. If mapper class or reducer class takes little effect in some step, it is ignored when we elaborate. The whole process is comprised of four steps; step 2 and step 3 are iterated for certain times or until convergence. Here are the details of each step as below.

Step 1: Initialize the input similarities using class called InitMapper and then output the similarity, responsibility and availability in the form of "i: **Flag** k value". Here, colon is used to separate the key and value in MapReduce framework. **Flag** is used to tell the type of the value, which could be one of the values in '**s**', '***r***', and '***a***', representing similarity, responsibility and availability correspondingly. In step 1, all the values of responsibility and availability are initialized to zero.

Step 2: Compute the "responsibilities" according to equation 2, and the output of responsibility is a little tricky, by which we mean, the output of "responsibility" and "availability" of step 2 is in the form of "*k*: **Flag** *i* value". For instance, $r(i, k)$=0.5 is represented as "*k*: ***r*** *i* 0.5", and similarly $a(i, k)$=0.1 is represented as "*k*: ***a*** *i* 0.1". The reason is shown in step 3. To calculate $r(i, k)$, all the values of $a(i, k')$ and $s(i, k')$ are needed as input. Though computation is distributed on different computers, all the values taking *i* as key are passed into the reduce function in class SA2RReducer as a list. In reduce function, it is easy to tell the type of the value according to the **Flag**, and to figure out the correct value of $r(i, k)$ after looking over all the values with *i* as key.

Step 3: Compute the "availabilities" according to equation 3 and equation 4, then output "responsibilities" and "availabilities" in normal order as the input of step 2 of the next iteration. Similarly to step 2, the calculation of $a(i, k)$ needs all the values of $r(i', k)$ as input. Since after step 2, all the key/value pairs are indexed by the column, all the values in the *k-th* column are organized as a list; it's possible to calculate $\sum_{i's.t.i'\notin\{i,k\}} \max\{0, r(i', k)\}$, which is part of equation 3 and equation 4.

Step 4: Figure out the exemplars for all the data points using the result of iteration of step 2 and step 3. First, reduce function in class CleanReducer finds the candidate exemplars whose summation of "availability" and "responsibility" is larger than zero. Second, for each data point *k*, it chooses from candidate exemplars with largest similarity to *k* as the real exemplar for *k*. It is worth noting that the task number of step 4 has to be set to 1, by which we mean this step can't be distributed. Because each choice of exemplar for data point can only be made after all the candidate exemplars are looked up, which makes the step can only be processed on the unique computer.

4 Experiments

To evaluate the effectiveness of our approaches, we experimented with 1425 e-learning documents. The 1425 documents are divided into 150 conceptual topics, and generally each conceptual topic contains less than 20 documents. In our experiments, we evaluate the precision of our method, which is defined as .:

$$precision = \frac{the\ \ number\ \ of\ \ correctly\ \ returned\ \ objects}{the\ \ number\ \ of\ \ total\ \ objects\ \ returned} \quad (5)$$

Besides, to evaluate how effective our approaches are to retrieve e-learning materials a user want to find, we define coverage as:

$$coverage = \frac{the\ \ number\ \ of\ \ correctly\ \ returned\ \ objects}{the\ \ number\ \ of\ \ relevant\ \ objects\ \ in\ \ database} \quad (6)$$

4.1 Conceptual Topic Clustering

In this section, we mainly evaluate the performance of the conceptual topic clustering of e-learning resources. In our experiment, each document is assigned to a conceptual topic in advance. After the unsupervised clustering by affinity propagation, every document in each cluster is looked up in the digital library and its real conceptual topic is found. Then, we assign the most common topic to the cluster, calculate the ratio of the number of documents in the cluster that belonging in the most common topic over the total number of documents in the cluster and regard it as a clustering accuracy percentage.

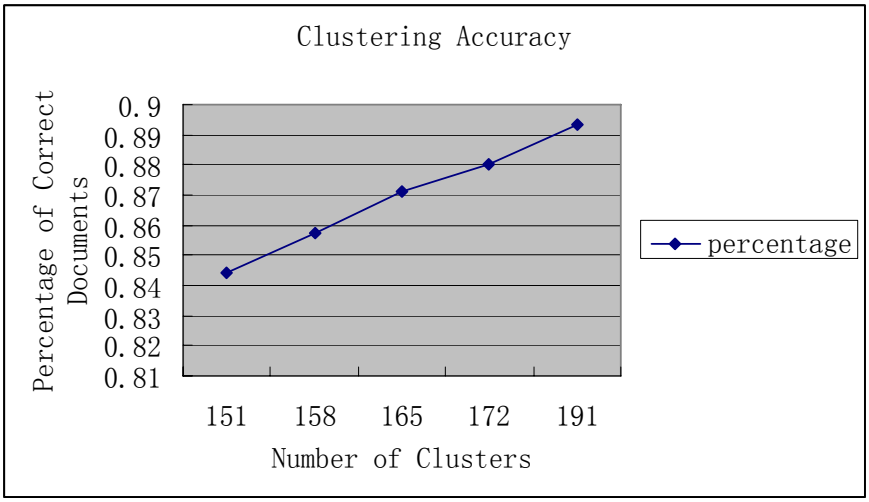

Fig. 2. The accuracy of clustering with different number of clusters.

Figure 2 shows the accuracy of clustering when the number of clusters is changed. From figure 2, we can see that the number of clusters produced by affinity propagation ranges from 151 to 191, and the accuracy percentage is around 0.87. The result of clustering is crucial to the final retrieval results, since by clustering, irrelevant documents are excluded and will not be searched. According to this experiment, we can see that less than 20 percent of irrelevant documents are clustered into the not so relevant topic, which could satisfy the need of excluding most irrelevant documents.

4.2 Query Relevant E-learning Resources

In this section, we do some experiments to test the performance of the proposed method. Before the retrieval with LSI, e-learning documents have been clustered into 165 conceptual topics. When a query is arrived, several target clusters are chosen to search from, which are determined according to similarities of query and exemplars. Obviously, if the query is a document in database, it will choose right the cluster it is clustered to as the first target cluster. The standard LSI is used to search the target

clusters then. In our experiment, for each query, top 24 results are returned. If the total number of documents in target clusters is less than 24, then all the documents in target clusters are returned. Usually, we return top several documents as results in each target cluster averagely.

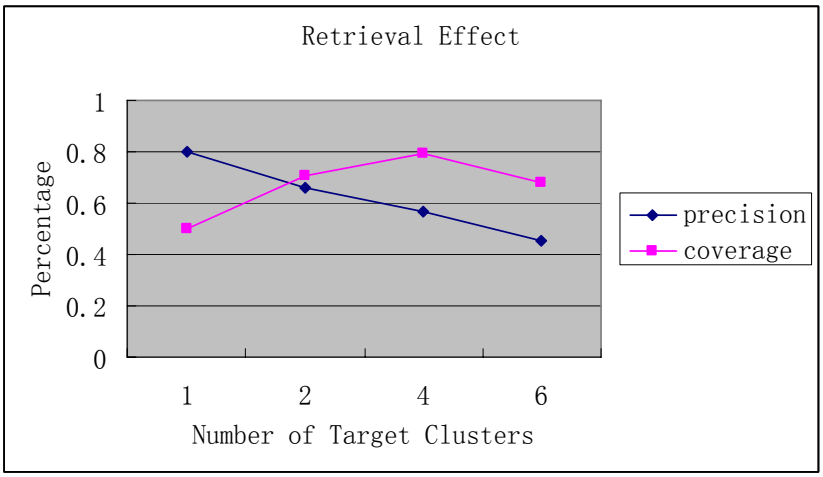

Fig. 3. The accuracy of retrieval with different number of target clusters.

Figure 3 shows the accuracy and coverage of proposed method. Figure 3 reflects the influence caused by number of target clusters. When only one cluster serves as the target cluster, the precision is best, since most of documents in the only cluster is relevant to the query; the coverage is not good enough though, because part of this conceptual topic is clustered into other clusters; besides, in some cases, the query semantically relates to several topics, which makes the converage low with only one target cluster. As the number of target clusters increase, the coverage grows up at first due to more target clusters are took into account, and the precision falls because more less relevant documents in new introduced clusters are returned as results. It is worth noting that when the number of target clusters grows to six, the coverage also falls relative to the situation when number of target clusters is four. This is caused when the total number of documents in all target clusters exceeds 24, then less results in the best several clusters could be returned. According to our experiments, we can see that over 50% of relevant documents are in one cluster, and we could choose two or three clusters as target clusters, taking coverage into consideration.

6. Conclusion

In this paper, we adapted affinity propagation in MapReduce framework which is implemented by project Hadoop to make the clustering method applicable to large-scale data, since the proposed parallel affinity propagation could run in distributed way. We also introduced a method to retrieve e-learning resources according to conceptual topics efficiently, utilizing the proposed parallel affinity propagation. Experiment shows this method retrieves relevant resources relatively accurate and

takes little time, which benefits from the off-line clustering limiting the target clusters to search from.

Acknowledgement

This work is supported by National Natural Science Foundation of China (No.60533090, No.60525108), Key Technology R&D Program(2006BAH02A13-4), The National High Technology Research and Development Program of China (2006AA010107), Program for Changjiang Scholars and Innovative Research Team in University ((IRT0652, PCSIRT)

References:

1. EC (2000). Communication from the Commission: E-Learning - Designing "Tejas at Niit" tomorrow's education. Brussels: European Commission
2. Berry, M. W., Dumais, S. T. and O'Brien, G.W, Using linear algebra for intelligent information retrieval. SIAM: Review, 37:573-595, 1995
3. Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., & Harshman, R, Indexing By Latent Semantic Analysis, Journal of the American Society For I n formation Science , 4(1): 391-407,1990
4. T. K. Landauer, P. Foltz, and D. Laham, Introduction to latent semantic analysis. Discourse Processes, 25,1998
5. Peter D. Turney: Measuring Semantic Similarity by Latent Relational Analysis. IJCAI 2005: 1136-1141
6. G. Salton and M.E. Lesk, Computer evaluation of indexing and text processing, pages 143–180. Prentice Hall, Ing. Englewood Cliffs, New Jersey, 1971
7. J. S. Kandola, J. Shawe-Taylor, and N. Cristianini, Learning Semantic Similarity, In Advances in Neural Information Processing Systems (NIPS) 15, pages 657--664, 2002
8. A. Budanitsky and G. Hirst., Semantic distance in wordnet:An experimental, application-oriented evaluation of five measures. In Proceedings of the NAACL Workshop on Word-Net and Other Lexical Resources, Pittsburgh, 2001.
9. J. Dean and S. Ghemawat. Mapreduce: Simplified data processing on large clusters. Operating Systems Design and Implementation, pages 137–149, 2004.
10. Brendan J. Frey and Delbert Dueck, Clustering by Passing Messages Between Data Points, Science, 315 (5814): 972-976,2007
11. V. P. Pauca, F. Shahnaz, M. W. Berry, R. J. Plemmons. Text mining using non-negative matrix factorization. In Proceedings of the 2004 SIAM International conference on Data Mining (SDM 2004), pages 452–456, Lake Buena Vista, Florida, 2004.
12. W. Xu, X. Liu, Y. Gong. Document clustering based on non-negative matrix factorization. In Proceedings of the 26th annual international ACM SIGIR, pages 267–273, Toronto, Canada, 2003.