Computer and Information Science
www.ccsenet.org/cis
Vol. 3, No. 1
February 2010

CLCL-A Clustering Algorithm Based on Lexical Chain for Large-Scale Documents

Ming Liu (Corresponding author)
School of Computer Science and Technology, Harbin Institute of Technology
PO box 319, Harbin, Heilongjiang Province, 150001, China
E-mail: mliu@insun.hit.edu.cn

Xiaolong Wang
School of Computer Science and Technology, Harbin Institute of Technology
PO box 319, Harbin, Heilongjiang Province, 150001, China
Tel: 86-451-8641-3322 E-mail: xlwang@insun.hit.edu.cn

Yuanchao Liu
School of Computer Science and Technology, Harbin Institute of Technology
PO box 319, Harbin, Heilongjiang Province, 150001, China
Tel: 86-451-8641-3322 E-mail: lyc@insun.hit.edu.cn

The research is financed by National Natural Science Foundation of China No.60435020, National High Technology Development 863 Program of China No. 2007AA01Z172. (Sponsoring information)

Abstract

Along with explosion of information, how to cluster large-scale documents has become more and more important. This paper proposes a novel document clustering algorithm (CLCL) to solve this problem. This algorithm first constructs lexical chains from feature space to reflect different topics which input documents contain, and documents also can be separated into clusters by these lexical chains. However, this separation is too rough. So, idea of self organizing mapping is used to optimize cluster partition. For agglomerating documents with semantic similarities into one cluster, influences from similar features are also considered. Experiments demonstrate that because effects of semantic similarities between different documents are considered, CLCL has better performance than traditional document clustering algorithms.

Keywords: Lexical chain, Large-scale document clustering, Self organizing mapping, Neuron adjustment

1. Introduction

Along with evolvement of technology on internet, how to cluster large-scale information from website has become more and more important. After some tags are removed from webpage which are coded as HTML or XML, information from website is just document. This is why document clustering has become an effective method to analyze information from website (Niu et al., 2007; Luo et al., 2009; Xu & Wunsch, 2005). Recently, there are many document clustering algorithms, such as K-means (Guo et al., 2006), FTC (Beil et al., 2002), FCM (Wang et al., 2004), Hierarchical clustering (Wu et al., 2009), WEBSOM (Azcarraga et al., 2004). These algorithms all use Vector Space Model (VSM) to organize documents. In this model, all the features from feature space are used to construct vectors to represent documents and clusters. This method not only imports many useless features, but also augments dimension number of vectors to increase running time. This problem is called as “disaster of dimensionality”. When dimension number of feature space becomes larger, this problem is worse. That will make similarities among most documents close to 0, and will decrease partition ability of document clustering algorithms (Jennifer & Carla, 2004).

There is another problem of traditional clustering algorithm. That is traditional algorithms often use Euclidean distance as similarity computation. This similarity is decided by minus between weights of same feature in different vectors (Tuomo et al., 2007). In document clustering, words are often used as features, and different words may have semantic similarity (Liu et al., 2009). That means different words reflect similar meaning. This situation will cause that even if feature vectors of different documents don't share same features, these two documents will also reflect similar meaning because of features which have semantic similarities among them. There have been proposed some clustering algorithms which import semantic similarity, such as ConSOM (Liu et al., 2008). It uses WordNet to compute semantic similarity between different features, and combine semantic similarity and Euclidean distance together to form a novel similarity computation method. However, it neglects importing semantic similarity to optimize partition of clusters.

In order to solve previous problems, a novel document clustering algorithm (CLCL) is proposed in this paper. This algorithm constructs lexical chains to remove features which are irrelative to topic of document. Lexical chains are also used to construct initial clusters. After construction of initial clusters, idea of self organizing mapping is imported to optimize partition of clusters and influences from similar features are considered in similarity computation and neuron adjustment. Experiments demonstrate that because semantic similarity is imported, precision and time complexity of CLCL are better than those of traditional document clustering algorithms.

2. Construction of lexical chain

Lexical chain is first proposed by Hirst in literature (Jane & Graeme, 1991). It is constructed by linearly scanning feature space, and each chain may reflect profile of topic information which document reflects. This technique has been applied in many fields, such as text analysis and abstract extraction (Chan, 2004; Kumar et al., 2003). In order to construct lexical chain, the first thing is to know how to compute similarity between different features. In this paper, co-occurring word vectors are constructed to compute semantic similarity between different features.

2.1 Semantic similarity computation

The linguist indicates: "the syntax function of a feature is the distribution of this feature" (Sven et al., 1998). The context of a feature is a typical distribution. So, if the contexts of different features are almost the same, the semantic similarity between these features is close. The word before the feature and the word after the feature mostly determine the semantic meaning of this feature. So, in this paper, co-occurring word and co-occurring word probability are used to construct feature's co-occurring word vector. Each dimensionality of this vector corresponds to one co-occurring word. The value of this dimensionality is the co-occurring probability between the feature and its co-occurring word. By computing the Kullback-Leibler divergence (Frans, 2005) between different co-occurring word vectors, semantic similarity between different features can be gotten.

$$SSim(F_p, F_q) = 1 - H(\boldsymbol{FV(F_p)}, \boldsymbol{FV(F_q)}) \tag{1}$$

Formula (1) describes how to compute similarity between tow features. $H(\boldsymbol{FV(F_p)}, \boldsymbol{FV(F_q)})$ is the Kullback-Leibler divergence between two co-occurring word vectors-$\boldsymbol{FV(F_p)}$ and $\boldsymbol{FV(F_q)}$-of features F_p and F_q. Main meaning of this formula can be gotten from formula (3).

$$P_p(CoW_k) = \frac{Fre(F_p, CoW_k)}{Fre(F_p) * Fre(CoW_k)} \tag{2}$$

Formula (2) shows how to compute co-occurring probability of feature-F_p and its co-occurring word-CoW_k. This probability can reflect the semantic relation between the feature and its co-occurring word at certain degree (Rishi & David, 2006).

$$H(\boldsymbol{FV(F_p)}, \boldsymbol{FV(F_q)}) = -\sum_{i=1}^{n} \frac{(p_i + q_i)}{2} \log_2(p_i + q_i) + \frac{1}{2}\left[\sum_{i=1}^{n}(p_i * \log_2 p_i) + \sum_{i=1}^{n}(q_i * \log_2 q_i)\right] \tag{3}$$

Formula (3) describes the Kullback-Leibler divergence between F_p and F_q. p_i is ith co-occurring word probability in $\boldsymbol{FV(F_p)}$, and q_i is ith co-occurring word probability in $\boldsymbol{FV(F_q)}$. n represents the size of co-occurring word vector. From formula (3), it can be gotten that, the larger the difference between the contexts of different features is, the bigger the value of Kullback-Leibler divergence is. So, Kullback-Leibler divergence can compute the semantic similarity between different features.

2.2 Construction of lexical chain

After semantic similarity is computed, lexical chains can be constructed to represent topic which each document reflects. Besides, lexical chains also can be used to separate feature space to construct initial clusters. In this method, each chain represents one cluster, and this cluster includes the documents which all reflect the topic which this chain describes. Construction steps of lexical chain are shown as follows.

[1] Assume input document set as D, and ith document in it as D_i. Assume FS_i as feature set of D_i. Assume LC_i as chain

set of D_i, and L_k as kth lexical chain in LC_i .

[2] Scan FS_i from top to down. Assume F_j as the feature which is being scanned.

[3] Use formula (4) to compute similarity between F_j and each lexical chain in LC_i. Assume L_k as the lexical chain which has the max similarity to F_j, and insert F_j in L_k.

[4] Repeat steps [1] ~ [3] until all the features in FS_i have been scanned.

$$Sim(L_k, F_j) = \sum_{LF_d \in L_k} SSim(LF_d, F_j) \Big/ |L_k| \tag{4}$$

Formula (4) shows how to compute similarity between lexical chain and feature. In this formula, LF_d means dth feature which is includes by L_k. This formula uses average similarity between F_j and each feature in L_k to be the similarity between L_k and F_j as literature (Gonenc and Ilyas, 2007) shows.

After previous construction, one set can be used to include lexical chains of each document, and each chain in it reflects one subtopic which this document describes. Formula (5) shows how to compute the weight of lexical chain L_k . By this formula, the lexical chain which has the largest weight can be regarded as the representation of topic which this document emphasizes.

$$w(L_k, D_i) = \sum_{LF_d \in L_k} w_d(LF_d, D_i) \times \log(|L_k|) \tag{5}$$

In formula (5), $w_d(LF_d,D_i)$ represents weight of LF_d in document D_i. This weight is computed by classical TF/IDF (Akiko, 2004) method as formula (6) shows.

$$w_d(LF_d, D_i) = fre(LF_d, D_i) \Big/ fre(LF_d) \tag{6}$$

In formula (6), $fre(LF_d,D_i)$ represents frequency of appearances of LF_d in D_i. It is TF value of LF_d. $fre(LF_d)$ represents number of documents which includes LF_d. It is DF value of LF_d.

Previous steps also can be used to partition feature space into some lexical chains. Each chain among them represents one cluster, and this cluster includes the documents which all reflect similar information to this chain. The approach which constructs lexical chains from feature space is similar to previous approach which constructs lexical chains from document. It is shown as follows.

[1] Assume input document set as D. Assume FS as feature space of D. Assume LS as lexical chain set of FS, and L_k as kth lexical chain in LS.

[2] Scan FS from top to down. Assume FS_j as the feature which is being scanned.

[3] Use formula (4) to compute similarity between F_j and each lexical chain in LS. Assume L_k as the lexical chain which has the max similarity to F_j, and insert F_j in L_k.

[4] Repeat steps [1] ~ [3] until all the features in FS have been scanned.

2.3 Initial cluster partition by lexical chain

After previous construction, each lexical chain can represent one cluster, and we can compute semantic similarity between document and lexical chain to map documents into clusters.

Formula (7) shows semantic similarity between document and cluster. In this formula, $L(D_i)$ represents lexical chain which is constructed from document D_i and has the largest weight. This chain is used as the representation of topic which D_i reflects. L_k represents lexical chain which is constructed to represent cluster C_k. $w_d(f_t,D_i)$ represents weight of f_t in document D_i as formula (6) shows. $w_c(f_t,C_k)$ represents weight of f_t in cluster C_k. It can be computed by formula (8).

In formula (7), two parameters are combined to compute similarity between document and cluster. They are intersection between lexical chains of document and cluster and weights of features in this intersection.

$$Sim(D_i, C_k) = Sim(L(D_i), L_k) = \frac{\sum_{f_t \in L(D_i) \&\& f_t \in L_k} w_d(f_t, D_i) + w_c(f_t, C_k)}{\sum_{f_p \in L(D_i)} w_d(f_p, D_i) + \sum_{f_q \in L_k} w_c(f_q, C_k)} \tag{7}$$

$$w_c(f_t, C_k) = \sum_{D_l \in C_k} \sum_{f_t \in L(D_l)} w_d(f_t, D_l) \Big/ |D_l| \tag{8}$$

In the following section, idea of self organizing mapping is imported to optimize cluster partition. From experiments, we can see, after lexical chains are used to get initial clusters, CLCL only needs little running time to get convergence. This is because, initial clusters, which are partitioned by lexical chains, include documents which reflect distinct topics. So, there are few documents which are not correctly partitioned, and it only needs few iterative steps to perform adjustments.

3. Training approach

Previous cluster partition is not exact. There are some documents which are not correctly partitioned. The one reason to this situation is that previous lexical chains are constructed linearly. They may include some features which reflect irrelative meanings. The other reason is that there are some features which have several different meanings. That means they will be included by more than one lexical chain, whereas, previous construction can't satisfy this situation.

In order to solve previous problems, idea of self organizing mapping (SOM) is imported. There are two parameters of SOM (Kohonen, 1997; Alahakoon & Halgamuge, 2000). They are neuron topology and initial neurons. In CLCL, each cluster which is constructed from lexical chain is set as one initial neuron. Assume N_k as the neuron which is constructed from lexical chain L_k. Features in N_k are separated as two parts. They are features which are included by L_k and features which aren't included by L_k. Weights of features which are included by L_k are set as their weights in L_k. Weights of features which aren't included by L_k are set as 0. After previous operations, initial neurons are gotten. In CLCL, neurons are organized as square topology as Figure 1 shows (Andreas et al., 2002). This topology has two layers. The upper layer is competitive layer. It includes neurons. The lower layer is input layer. It includes vectors of input documents.

SOM and its varieties often use the following approach to perform clustering.

[1] Initialize neuron topology of SOM. Evaluate each dimensionality of each neuron vector with a small random value.

[2] Randomly select a datum as input of SOM network. Assume this datum as D_k.

[3] Use formula (1) to compute similarity between D_k and each neuron in neuron structure. The neuron which has the max similarity is the winner.

[4] Use monotonous anneal algorithm to adjust vector of winner neuron and other vectors of neurons which are in the neighborhood of winner neuron.

[5] Estimate whether clustering algorithm achieves convergence condition or not. If achieve, stop running. If not, repeat [2] ~ [5] until it achieves convergence condition.

Traditional SOM algorithms use Euclidean distance as similarity computation method (Kohonen et al., 2000). Only the weights of features which are both included by neuron and document are considered by this method. However, it neglects semantic similarity between documents. That means documents which don't share same features may reflect similar meanings. This is caused by some features which have semantic similarity between them, such as "internet" and "network". If they are respectively included by different documents, these documents may reflect similar meanings.

Semantic similarity is considered in CLCL. When one feature in neuron is adjusted, the features which reflect similar meaning to it are also adjusted. If previous method isn't performed, the documents, which have semantic similarities, may be partitioned into different clusters. This assumption is proved in the following paragraph.

Assume features f_1 and f_2 have semantic similarity, such as "internet" and "network". It is possible to find two initial neurons, such as n_1 and n_2. Weight of f_1 is large in n_1, and weight of f_2 is large in n_2. Assume d_1 and d_2 are two documents which respectively include f_1 and f_2. If documents d_1 and d_2 are selected to adjust neurons in training approach, d_1 will map to n_1 and weight of f_1 is adjusted larger to make d_1 and n_1 more similar. Certainly, d_2 will map to n_2 and weight of f_2 is adjusted larger to make d_2 and n_2 more similar. When algorithm converges, weight of f_1 will be large in n_1, and weight of f_2 will be large in n_2. This situation makes the gap between weights of f_1 and f_2 augment in the same neuron. As literature (Kohonen, 1997) shows, if feature in neuron has larger weight, the documents which are mapped to this neuron will reflect similar meaning to this feature. We know f_1 represents "internet", and f_2 represents "network". So, the documents which map to n_1 reflect similar meaning to "internet" and the documents which map to n_2 reflect similar meaning to "network". Because "internet" and "network" reflect similar meaning, the clusters which are gotten from n_1 and n_2 include the documents which reflect similar meanings. That means previous method partitions similar documents into different clusters.

In order to solve previous problem, weights of similar features are considered in similarity computation and neuron adjustment in training approach. From section 2, we know, lexical chain includes the features which reflect relative meanings. So, in CLCL, the features which are in the same lexical chain are regarded as similar features.

Because pervious method adjusts more features in training approach than traditional SOM algorithms. That will obviously reduce number of adjustment steps to get convergence, and also reduces clustering time. Figure 2 in experiments also proved it.

3.1 Similarity computation

Formula (9) shows how to adjust neuron vector according to input document. This formula considers influences from similar features on similarity computation.

$$Dist(D_i, N_k) = \begin{cases} \sum_{f_t} (W(f_t, D_i) - W(f_t, N_k))^2 & If\ f_t \in D_i; \\ \sum_{f_s} (SSim(f_t, f_s) \times W(f_t, D_i) - W(f_s, N_k))^2 & \\ \quad If\ f_s \notin D_i \,\&\&\, f_t \in D_i \,\&\&\, \exists p, f_s, f_t \in L_p; & \\ \sum_{f_s} \left(\frac{\sum_{t=1}^{m} SSim(f_t, f_s) \times W(f_t, D_i)}{m} - W(f_s, N_k) \right)^2 & \\ \quad If\ f_s \notin D_i \,\&\&\, f_t \in D_i (t = 1 \sim m) \,\&\&\, \exists p, f_s, f_t \in L_p; & \end{cases} \tag{9}$$

Previous formula has three parts. The first part is the same as Euclidean distance to compute similarity between document and neuron. This part disposes the features which appear in the document, such as feature f_t in the document D_i. The second part disposes the features which don't appear in the document, whereas, there are other features which appear in the document and they have semantic similarity to the features which don't appear in the document. Let's make it clearer. Assume f_s as the feature which doesn't appear in D_i. Assume f_t as the feature which appears in D_i, and it has semantic similarity to f_s. That also means f_t and f_s in the same lexical chain such as L_p. The second part of formula (9) uses semantic similarity between f_t and f_s to import influences from similar features on adjustment. There is another problem in previous adjustment. That is it exists more than one feature which appears in D_i and has semantic similarity to f_s. Assume these features as $f_1, f_2, \ldots, f_m$. The third part of formula (9) disposes this situation. This part uses average semantic similarity among features to perform adjustment.

3.2 Neuron adjustment

Formula (10) shows neuron adjustment according to input document. This adjustment is similar to previous similarity computation. It not only considers the features which appear in the document D_i such as f_t, but also considers the features which don't appear in the document D_i but have semantic similarity to f_t such as f_s.

$$\begin{cases} W(f_t, N_k)(t+1) = W(f_t, N_k)(t) + a(t)h(t)(W(f_t, D_i) - W(f_t, N_k)(t)) \\ \quad If\ f_t \in D_i; \\ W(f_s, N_k)(t+1) = W(f_s, N_k)(t) + a(t)h(t)(SSim(f_t, f_s) \times W(f_t, D_i) - W(f_s, N_k)(t)) \\ \quad If\ f_s \notin D_i \,\&\&\, f_t \in D_i \,\&\&\, \exists p, f_s, f_t \in L_p; \\ W(f_s, N_k)(t+1) = W(f_s, N_k)(t) + a(t)h(t)\left(\frac{\sum_{t=1}^{m} SSim(f_t, f_s) \times W(f_t, D_i)}{m} - W(f_s, N_k)(t) \right) \\ \quad If\ f_s \notin D_i \,\&\&\, f_t \in D_i (t = 1 \sim m) \,\&\&\, \exists p, f_s, f_t \in L_p; \end{cases} \tag{10}$$

Formula (10) also has three parts to adjust weights of features. The first part adjusts weights of features as traditional SOM algorithm. In this formula, $a(t)$ is learning rate, which decreases along with training approach. $h(t)$ is adjusted function, and Gauss function is often used as adjusted function (Kohonen et al., 2000). The second part is used to adjust f_s which doesn't appear in the document D_i but has semantic similarity to f_t. The third part disposes the situation that there is more than one feature which appears in D_i but has semantic similarity to f_s.

4. Experiments and analysis

Ten hundred thousand articles are selected from China Daily of 1998 as the first testing corpus. However, this corpus is too large, and each document from it can't be manually marked with certain cluster index. Precision and recall also can't be computed in this situation. So, a smaller corpus is constructed. Five thousand articles are randomly selected from website Yahoo as the second testing corpus. This corpus is manually classified into thirty classes. They include sports, entertainment, medicine, education, military, and so on.

In this paper, purity is used to test clustering precision on the smaller corpus (Gu et al., 2001).

$$P(S_r) = \frac{1}{n_r} \max_{q=1}^{z}(n_r^q) \tag{11}$$

In this formula, z represents cluster number or neuron number. S_r represents rth cluster after clustering algorithm. n_r represents data number of S_r. We know each datum in the smaller testing corpus is already marked with certain cluster index. Then, C_q is used to represent the cluster which includes the documents that are marked with qth cluster index in testing corpus. n^q is used to represent the number of documents in C_q. n_r^q is used to represent the number of documents, which belong to C_q in testing corpus and belong to S_r after clustering algorithm.

The average purity of different clusters can be used to represent precision of clustering algorithm.

$$Purity = \sum_{r=1}^{z} \frac{n_r}{n} P(S_r) \tag{12}$$

In Table 1, the performance of algorithm which imports semantic similarity in similarity computation and neuron adjustment is tested. In this table, we call the algorithm, which imports semantic similarity in similarity computation and neuron adjustment, as CLCL_S. We call the algorithm, which doesn't import semantic similarity, as CLCL_N. Table 1 shows clustering precision and running time of CLCL_S and CLCL_N on smaller corpus.

From Table 1, we can see, running time and clustering precision of CLCL_S are better than those of CLCL_N. The reason is that the clusters which are constructed from CLCL_S can agglomerate the documents which have semantic similarities into one cluster. This situation will obviously improve precision. Besides, when CLCL_S performs neuron adjustment, more features are adjusted. So, it obviously needs smaller iterative steps to get convergence.

Because the larger testing corpus is too large, we can't manually separate it and then compute precision and recall. So, we regard the clusters which are generated from clustering algorithm as C_1, C_2,…,C_m, and each cluster among them is separated into some sub-clusters by their meanings. Assume the sub-clusters which are separated from C_i as SC_{i1}, SC_{i2},…,SC_{iz}. The sub-cluster which has the max number of documents is used to represent C_i, and this sub-cluster is assumed as SC_{imax}. Regard other sub-clusters are irrelative to C_i. Then, clustering precision on C_i can be represented as: $|SC_{imax}|/|C_i|$. $|C_i|$ is the number of documents in C_i. The average precision on all the clusters is used to represent precision of clustering algorithm.

Running time and clustering precision of CLCL-S on larger testing corpus is respectively compared with those of CLCL-N in Figure 2 and Figure 3.

From Figure 2 and Figure 3, we can see that the method which imports semantic similarity in similarity computation and neuron adjustment can effectively improve clustering precision and reduce running time. When number of documents increases, this situation is more obvious. The reason is that, when number of documents increases, there are more features which have semantic similarities. So, there are more documents which have semantic similarities, and CLCL_S can better cluster them than CLCL_N. Besides, when more features have semantic similarities, CLCL_S needs to adjust more features in neuron adjustment. That will greatly decrease number of iterative steps, and will reduce running time.

Lexical chains which are constructed by linearly scanning feature space are shown in Table 2.

From Table 2, we can see, different lexical chains reflect distinct meanings. So, initial clusters which are constructed from these chains will have clear inter-cluster distinctness. However, there are some irrelative features in lexical chains which will decrease clustering precision, such as "city" in lexical chain 1 and "brand" in lexical chain 3.These features will be adjusted by training approach of CLCL.

Clustering precision and running time of different algorithms on smaller testing corpus are shown in Table 3. They include K-means, FTC, Hierarchical clustering, WEBSOM, ConSOM.

From Table 3, we can see, CLCL has the least running time among all the testing algorithms. This is because CLCL uses semantic similarity to construct lexical chains to partition documents into initial clusters. This partition is close to convergent partition. So, it needs few iterative steps to get convergence. This is one reason why CLCL has the lowest time complexity. In training approach of CLCL, similar features are also adjusted. This method increases number of features which are adjusted in training approach, and will decrease number of iterative steps. This is the other reason why CLCL has the lowest time complexity. Besides, similar features are considered in training approach. That will make cluster agglomerate documents which have semantic similarities, and will increase clustering precision greatly. From Table 3, we also can see, two algorithms based on SOM (WEBSOM and ConSOM) have better performance on time complexity and clustering precision. This is because, SOM uses neurons to represent clusters, and iteratively adjusts neuron vector to get convergence. Between these two algorithms, ConSOM has better precision. This is because ConSOM imports semantic similarity (Liu et al., 2008). However, this algorithm neglects influences from similar features on neuron adjustment. So, precision of ConSOM is lower than that of CLCL. Because, ConSOM has more steps to compute semantic similarity than WEBSOM, running time of WEBSOM is lower than that of ConSOM. FTC measures overlapping scale between feature sets to compute similarity between different documents (Beil et al., 2002). This computation certainly can't find semantic similarity between different documents. Besides, FTC doesn't perform any optimization on cluster partition. So, precision of FTC is little lower. FTC needs to linearly scan feature space to cluster documents. So, its time complexity is linear with the scale of feature space. In our experiments, running time of FTC is in the middle position among testing clustering algorithms. Running time of K-means is short and it is close to CLCL. This is because K-means has linear time complexity. However, because K-means uses Euclidean distance to compute similarity, precision of K-means is lower. Time complexity and clustering precision of Hierarchical clustering are both worst. This is because it has $O(n^2)$ time complexity. Besides, when cluster is formed, documents which belong to this cluster can't be moved to different clusters.

Running time and clustering precision of CLCL are also compared with those of other clustering algorithms on larger testing corpus. They are shown in Figure 4 and Figure 5.

From Figure 4, we can see, precisions of the algorithms (CLCL, ConSOM, and WEBSOM) are stable, when number of documents increases. This is because these algorithms are based on the idea of self organizing mapping. They map high dimensional space into low dimensional plane, which can avoid the interferences from the problem of "disaster of dimensionality". We also can see that rank of precisions from low to high is WEBSOM, ConSOM and CLCL. This is because semantic similarity is considered in CLCL and ConSOM. So, precisions of them are better than that of WEBSOM. In comparison with ConSOM, CLCL not only imports semantic similarity but also considers influences from similar features in adjustment approach. So, precision of CLCL is better than that of ConSOM. Precision of FTC is not greatly affected by number of documents. This is because FTC uses overlapping scale between feature sets to measure similarity. This method can avoid the problem aroused by high dimension number. Precisions of K-means and Hierarchical clustering drop greatly, when number of documents increase. This is because they don't perform any optimization to cluster documents in high dimensional space. So, when number of documents increases, high dimensional feature space will greatly decrease clustering precision.

From Figure 5, we can see, when number of documents increases, running time of CLCL, ConSOM, and WEBSOM increases stably. This is because they map high dimensional feature space to low dimensional plane. This method can avoid interferences aroused by high dimension number. When number of documents increases, running time of FTC increases greatly. This is because, time complexity of FTC is sensitive to dimension number of feature space. Running time of K-means and Hierarchical clustering also increases greatly, when number of documents increases. This is because they use Vector Space Model (VSM) to organize documents. When number of documents increases, it needs much time to compute similarity by this model.

5. Conclusions

A novel clustering algorithm based on lexical chain (CLCL) is proposed in this paper for large-scale documents. This algorithm first constructs co-occurring word vectors to compute semantic similarity between different features. After computation, lexical chains are constructed to partition documents into some initial clusters. Experiments demonstrate that these initial clusters reflect distinct meanings and need few iterative steps to get convergence. In order to agglomerate documents which have semantic similarities into one cluster, similar features are considered in similarity computation and neuron adjustment. This method not only can improve clustering precision but also can reduce running time. Experiments demonstrate running time and clustering precision of CLCL are both better than those of traditional clustering algorithms.

References

Akiko A. (2004). An information-theoretic perspective of tf–idf measures. *Information Processing and Management*, 39, 45-65.

Alahakoon D, & Halgamuge S. K. (2000). Dynamic self-organizing maps with controlled growth for knowledge discovery. *IEEE Transactions on Neural Networks*, 11, 601-614.

Alam H. Kumar, A. Nakamura, M. Rahman, F. Tarnikova, & Y. Che Wilcox. (2003). Structured and unstructured document summarization: design of a commercial summarizer using Lexical chains. *Proceedings of the Seventh International Conference on Document Analysis and Recognition*, pp. 1147-1152.

Andreas Rauber, Dieter Merkl, & Michael Dittenbach. (2002). The growing hierarchical self-organizing map: exploratory analysis of high-dimensional data. *IEEE Transactions on Neural Networks*, 13, 1331-1341.

Arnulfo P. Azcarraga, Teddy N. Yap, Jonathan Tan, & Tat Seng Chua. (2004). Evaluating keyword selection methods for WEBSOM text archives. *IEEE Transactions on Knowledge and Data Engineering*, 16, 380-383.

Chan S. W. (2004). Extraction of salient textual patterns: synergy between lexical cohesion and contextual coherence. *IEEE Transactions on Systems, Man, and Cybernetics. Part A, Systems and Humans*, 34, 205-218.

Congnan Luo, Yanjun Li, & Soon M. Chung. (2009). Text document clustering based on neighbors. *Data and Knowledge Engineering*, 68, 1271-1288.

Florian Beil, Martin Ester, & Xiaowei Xu. (2002). Frequent term-based text clustering. *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 436-442.

Frans M. Coetzee. (2005). Correcting the Kullback–Leibler distance for feature selection. *Pattern Recognition Letters*, 26, 1675-1683.

Gonenc E, & Ilyas C. (2007). Using lexical chains for keyword extraction. *Information Processing and Management*, 43, 1705-1714.

Hai-xiang Guo, Ke-jun Zhu, Si-wei Gao, & PTing Liu. (2006). An improved genetic k-means algorithm for optimal clustering. *Proceedings of the Sixth IEEE International Conference on Data Mining*, pp. 793-797.

Jane Morris, & Graeme Hirst. (1991). Lexical cohesion computed by thesaural relations as an indicator of the structure of text. *Computational Linguistics*, 17, 21-48.

Jennifer G. Dy, & Carla E. Brodley. (2004). Feature selection for unsupervised learning. *Journal of Machine Learning Research*, 5, 845–889.

Junjie Wu, Hui Xiong, & Jian Chen. (2009). Towards understanding hierarchical clustering: a data distribution perspective. *Neurocomputing*, 72, 2319-2330.

M. Gu, H. Zha, C. Ding, X. He, H. Simon, & J. Xia. (2001). *Spectral relaxation models and structure analysis for k-way graph clustering and bi-clustering*. Technical Report, CSE-01-007, Penn State University.

Min Liu, Weiming Shen, Qi Hao, & Junwei Yan. (2009). A weighted ontology-based semantic similarity algorithm for web service. *Expert Systems with Applications*, 36, 12480-12490.

Rui Xu, & Wunsch D. II. (2005). Survey of clustering algorithms. *IEEE Transactions on Neural Networks*, 16, 645-678.

Rishi Jobanputra, & David A. Clausi. (2006). Preserving boundaries for image texture segmentation using grey level co-occurring probabilities. *Pattern Recognition*, 39, 234-245.

Sven M., Jorg L., & Hermann N. (1998). Algorithms for bigram and trigram word clustering. *Speech Communication*, 24, 19-37.

T. kohonen. (1997). *Self-organizing maps*. Berlin: Springer, (Chapter 3).

Teuvo Kohonen, Samuel Kaski, Krista Lagus, Jarkko Salojarvi, Vesa Paatero, & Antti Saarela. (2000). Self organization of a massive document collection. *IEEE Transactions on Neural Networks*, 11, 574-585.

Tuomo Korenius, Jorma Laurikkala, & Martti Juhola. (2007). On principal component analysis, cosine and Euclidean measures in information retrieval. *Information Sciences*, 177, 4893-4905.

X. Z. Wang, Y. D. Wang, & L. J. Wang. (2004). Improving fuzzy c-means clustering based on feature weight learning. *Pattern Recognition Letters*, 25, 1123-1132.

Yuanchao Liu, Xiaolong Wang, & Chong Wu. (2008). ConSOM: A conceptional self-organizing map model for text clustering. *Neurocomputing*, 71, 857-862.

Zheng-Yu Niu, Dong-Hong Ji, & Chew Lim Tan. (2007). Using cluster validation criterion to identify optimal feature subset and cluster number for document clustering. *Information Processing and Management*, 43, 730-739.

Table 1. Clustering time and clustering precision of CLCL_S and CLCL_N on smaller corpus

Methods	CLCL_S	CLCL_N
Time (s)	83	127
Precision (%)	84.13	77.36

Table 2. Lexical chains constructed by linearly scanning feature space

List	Feature chains
1	champion, competition, game, train, match, almightiness, athlete, Olympic game, judgment, score, goal, coach, club, world cup, city, torch, gym, race, trials, contest,
2	economy, price, commodity, company, technique, garden, enterprise, industry, agriculture, labor, product, supply, order goods, trade, market, shop, check,
3	computer, crash, PC, machine, cursor, hypertext, pure text, icon, link, space, brand, byte, internet, off-line, memory, website, notebook, processor, keyboard, mouse,
4	land, earth, location, mountain, sea, coast, port, delta, furrow, bog, highland, shore, seaport, dock, cattle farm, cliff, peak, river bank, grotto, latitude, geography,
5	professor, research, investigate, teacher, doctor, bachelor, student, building, grade, school, certification, education, diploma, suspend classes, dissertation, college,

Table 3. Clustering precision and running time of different algorithms

Methods	K-means	FTC	Hierarchical	WEBSOM	ConSOM	CLCL
Time (s)	85	176	273	126	149	83
Precision (%)	69.51	73.07	68.49	76.41	79.98	84.13

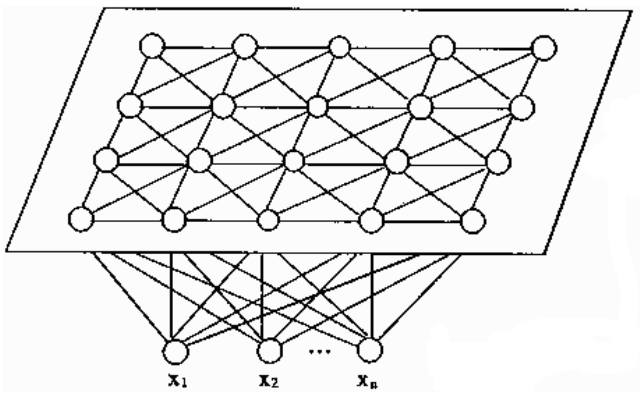

Figure 1. Square neuron topology of SOM

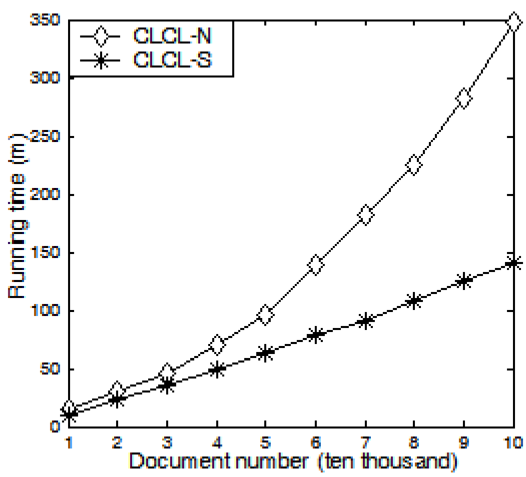

Figure 2. Running time of CLCL_S and CLCL_N

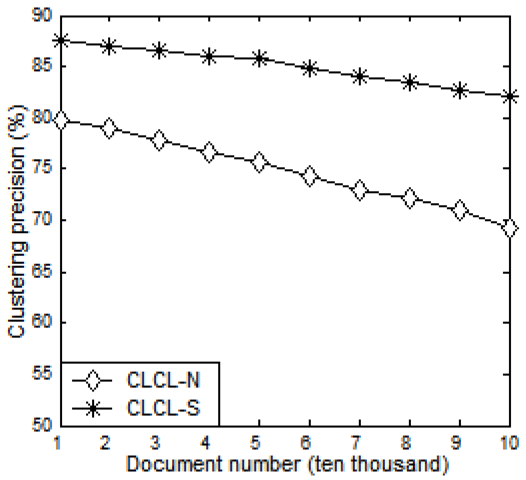

Figure 3. Clustering precisions of CLCL_S and CLCL_N

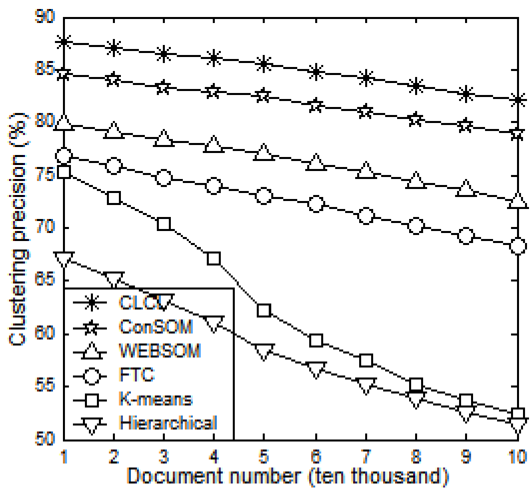

Figure 4. Clustering precisions of different algorithms

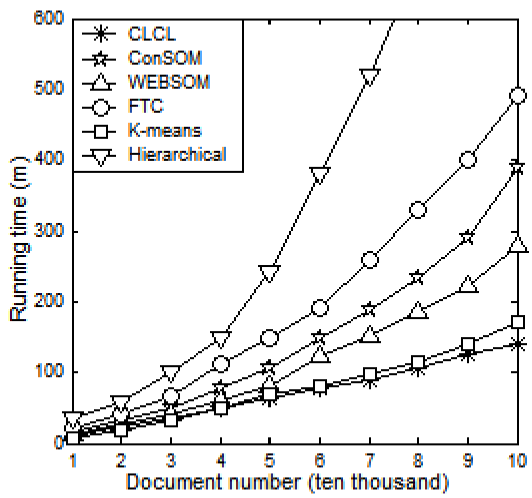

Figure 5. Running time of different algorithms