Core-Tag Clustering for Web 2.0 Based on Multi-similarity Measurements*

Yexi Jiang, Changjie Tang, Kaikuo Xu, Lei Duan, Liang Tang, Jie Gong, and Chuan Li

School of Computer Science Sichuan University, Chengdu, 610065 China
{jiangyexi,tangchangjie}@cs.scu.edu.cn

Abstract. Along with the development of Web2.0, folksonomy has become a hot topic related to data mining, information retrieval and social network. The tag semantic is the key for deep understanding the correlation of objects in folksonomy. This paper proposes two methods to cluster tags for core-tag by fusing multi-similarity measurements. The contributions of this paper include: (1) Proposing the concept of core-tag and the model of core-tag clusters. (2) Designing a core-tag clustering algorithm CETClustering, based on clustering ensemble method. (3) Designing a second kind of core-tag clustering algorithm named SkyTagClustering, based on skyline operator. (4) Comparing the two algorithms with modified K-means. Experiments show that the two algorithms are better than modified K-means with 20-30% on efficiency and 20% higher scores on quality.

Keyword: folksonomy, tag, clustering, clustering ensemble, skyline.

1 Introduction

With development of Web information technology, the available resources have increased dramatically. *Taxonomy* is used to classify the online resources. However, in most cases of Taxonomy, resources are categorized by experts, thus they cannot reflect the original opinions of the users. To solve these problems, the authors of [1, 2] proposed new concept named *folksonomy*. It allows users to add metadata in the form of keywords to shared resources. Folksonomy [3] allows users effectively organize and share vast amount of information. Intuitively, folksonomy is a tripartite graph, in which users, tags and resources are nodes while relations among them are edges. It reflects users' true opinions on resources via collaborative intelligence. The core of folksonomy is defining tags by users. Since it would cause a large number of different tags for similar resources, an efficient clustering mechanism to put tags together is necessary to get related tags by giving query tag. We call the query tag **core-tag**. In this paper, we focus on handling core-tag clustering. Example 1 illustrates the idea of core-tag clustering.

* Supported by the 11th Five Years Key Programs for Sci. & Tech. Development of China under grant No. 2006BAI05A01, the National Science Foundation under grant No. 60773169, the Software Innovation Project of Sichuan Youth under grant No. 2007AA0155.

L. Chen et al. (Eds.): APWeb and WAIM 2009, LNCS 5731, pp. 222–233, 2009.

Example 1. A user wants to find out the related tags of 'web2.0' and tries to know their closeness to the core tag but without trivial details. Suppose the related tag set be {blog, Social, library2.0, Design, Web, community, collaboration, mashup, online, webdesign, socialnetworking, video, education, ajax, aggregator, search techonology, Blogs, wiki, rss}. The clustering result can be seen in Table 1 and Fig. 1.

Table 1. Result of Core-tag Clustering

Clustering result by using core-tag 'web2.0'			
Group 1:	blog, Social, library2.0;	Group 2:	Design, Web, community;
Group 3:	collaboration, mashup;	Group 4:	online, webdesign, video;
Group 5:	education, aggregator;	Group 6:	search, technology, Blogs, wiki;

Clearly, the tags in a group are related with each other. Moreover, we can see that the group with smaller index is closer (to the core tag) than the group with larger index. The distance between a cluster and the core tag indicates their semantic similarity.

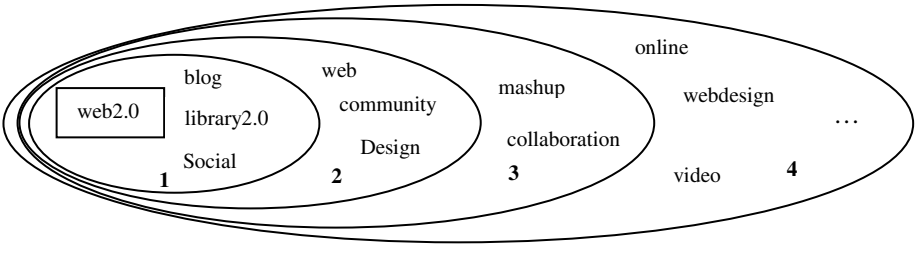

Fig. 1. Core-tag Clustering using core-tag

It is a challenging task with following difficulties:

(a) The existing association rule mining algorithms can find the correlated tags, but cannot efficiently group the correlated tags.
(b) Some existing clustering algorithms work well to find the groups, but clusters are unordered and cannot be distinguished.
(c) Some algorithms like K-means are not stable. They lead to the randomness of result. For different runs the result may be different. The randomness would lower the accuracy of the result.
(d) Some algorithms require a pre-fix parameter to set the number of clusters. It depends much on the subjective of the users. Such parameter settings are usually empirically set and difficult to be determined.

To overcome above problems, we introduce two core-tag clustering methods with different ideas. That is: (a) The idea of clustering ensemble and (b) the idea of sky-line. A formal definition of core-tag clustering is as follows:

Problem Statement. Given a core-tag T_c and set $\boldsymbol{S}$ that contains candidate tags related to T_c. Find a set of clusters contain at least one of the most related tags, let $\boldsymbol{C}$ denotes the set of clusters. The result should satisfy the following conditions:

- $\cup \boldsymbol{C} \subseteq \boldsymbol{S}$ and $|\cup \boldsymbol{C}| \leq k$. Variable k denotes the cardinality of $\boldsymbol{S}$.
- For $\forall \boldsymbol{C}_j \in \boldsymbol{C}$, there have $\boldsymbol{C}_j \neq \varnothing$ and $\boldsymbol{C} = \boldsymbol{C}_1 \cup \boldsymbol{C}_2 \ldots \cup \boldsymbol{C}_n$. Variable $\boldsymbol{C}_i$ denotes a certain cluster.
- For $\forall \boldsymbol{C}_i, \boldsymbol{C}_j \in \boldsymbol{C}$, there have either $\text{dist}(T_c, \boldsymbol{C}_i) < \text{dist}(T_c, \boldsymbol{C}_j)$ or $\text{dist}(T_c, \boldsymbol{C}_i) > \text{dist}(T_c, \boldsymbol{C}_j)$. The function dist() denotes the distance from the core-tag to a cluster by considering the distance of all tags in the cluster.

The contributions of this paper include: (1) Proposing the concept of core-tag and the model of core-tag clustering based on multi-similarity measurements. (2) Designing core-tag clustering algorithm CETClustering based on clustering ensemble method. (3) Designing core-tag cluster algorithm SkyTagClutering based on skyline operator. (4) Analyzing the best method based on experiment results both on the efficiency and the quality of the clustering result.

The rest of the paper is organized as follows: Section 2 describes the related work. Section 3 proposes algorithm CETClustering, based on clustering ensemble method. Section 4 proposes algorithm SkyTagClustering, based on skyline operator. Section 5 designs experiments to evaluate the two methods and compares them with K-means on efficiency and quality. Section 6 concludes the paper with a discussion and introduces the future work.

2 Related Work

Ensemble method was proposed for data fusion problem in the realm of information retrieval [6]. Then it was employed to improve the quality of decision tree in Quinlan's paper [8]. This method was widely applied in classifying. Literature [9], [10] and [11] introduced how this method be used in clustering field. N.C. Oza [9] gave an example. It used a set of K-means algorithm as basic algorithms and used the ensemble method to combine the results of these basic algorithms. A.Strehl and J.Ghosh proposed a framework of a classical clustering ensemble process in [10]. The key idea is as follows: (a) Clustering the data independently by a set of clusterers. (b) Constructing a graph based on the result. (c) Decomposing the graph into proper number of parts to get final result. In [11] the authors gave an experiment to analyze the quality of the method by counting the mis-assigned objects.

Consider the skyline operator method. Top-k problem is a classical problem in database field. It first appeared in S. Borzsony's paper [12]. It proposed a complicated but not so efficient algorithm for calculating the skyline. Literature [13] introduced two progressive algorithms to compute the skyline: The Bitmap version and the Index version. The Bitmap version is efficient for computing the skyline with discrete values. The Index version used a B-tree to help store the information and sort the element in each dimension to help get the skyline. They are efficient; however, they both need some prerequisite, such as: (a) Skyline operation involves all the dimensions of the data. (b) All the values are within the range [0, 1]. (c) Each dimension has discrete value. Unfortunately, the tag similarity datasets can only fulfill the first two requirements even after the preprocessing. For each dimension in our similarity set that represents the result of a certain similarity method, the value of the similarity is continuous and has the accuracy of 4 digit after the point, thus if the algorithm in [13] is

employed, there would be thousands of distinct values, which is prohibitive. In [14] and [15] the authors mainly focused on computing the skyline when elements has high dimension. For [14], it introduced the concept skyline frequency that compares and ranks the interestingness of data points based on how often they are returned in the skyline when different number of dimensions. Literature [15] introduced another new concept k-dominant. Their proposal of k-dominance offers a different notion of interestingness from skyline frequency. It focused on how many dimension an element E1 dominate E2. Authors of [16] developed BBS (Branch and Bound Skyline), a progressive algorithm based on nearest neighbor search, which is IO optimal. It employs an R-tree to help store the needed information and avoid the duplicates. The skyline algorithm in [17] can quickly return the first result and produce more and more results continuously. That means the algorithm can get part of the skyline before the whole process of computation.

3 The Algorithm CETClustering

In order to solve core-tag clustering problems, we propose an algorithm based on the membership of tags in the same cluster. [10] introduces a framework for combining multiple partitions of multiple clusterers by using the graph partitioning method. We use the same framework in our work. However, the graph partition methods they used are not suitable for the core-tag clustering, we use our own partition method by modify minimum spanning tree to maximum spanning forest. The reasons lies that: (a) The methods can only solve bi-partitioning problem. In tag clustering situation, it needs to partition the graph into any number of parts. (b) They aim to partition huge graph while in tag clustering problem the graph is much smaller.

The key steps of core-tag clustering are as follows:

Step 1: Use a number of clusterers to cluster tags independently.
Step 2: Generate the co-association matrix CoA according to the clustering result of each clusterers.
Step 3: Decompose the graph represented by the CoA to any number of parts wanted.

3.1 Preprocess

Before using CETClustering algorithm, we need to do some preprocess to the data in order to fulfill the input requirement.

In [4], we measured the similarity between the tags by eight formulas (AEMI, Simrank, etc.) Since there is no existing similarity measurement considering all the three factors altogether (user, tagged page and tag), we need to consider multiple measurements as a whole to cluster tags to improve the clustering result.

Table 2. Similarity Set

Java	Dimension 1	Dimension 2
J2EE	0.841	0.718
JVM	0.812	0.802
JDK	0.794	0.811
JRE	0.748	0
J2ME	0	0.718

The steps of preprocessing are as follows:

(1) Generate similarity set ***Set*** for each core-tag and remove unqualified elements. In this step, we have a set ***S*** for each core-tag T_c. It contains k n-dimensional vectors. The variable k is the cardinality of ***S***; n is the number of similarity measurements used. Each dimension of a vector denotes the similarity value between T_c and a certain related tag calculated by a certain measurement (See Table 2). We remove those vectors with having missing value in more than half dimensions and fill remain missing values with 0. Now the similarity set ***Set*** is created.
(2) Normalize all the values in set ***Set***. In this step we adopt min-max normalization to convert all the values into domain [0, 1].
(3) Fill in missing value in the set. The remaining missing values are filled with the average value of the other element in the vector.

	JDK	JRE	JVM	J2EE	J2ME	Eclipse	EJB
JDK	3	2	0	2	1	0	0
JRE	2	3	3	1	1	0	1
JVM	0	3	3	1	1	2	1
J2EE	2	1	1	3	2	2	3
J2ME	1	1	1	2	3	1	0
Eclipse	0	0	2	2	1	3	1
EJB	0	1	1	3	0	1	3

Fig. 2. Co-Association Matrix

3.2 Independent Clustering and Generate Co-association Matrix

In this step, we use several clusterers to cluster tags independently according to ***Set***. Here we use K-means as clusterer.

Co-Association matrix CoA (Fig. 2) represents the relationship between the tags based on the results of several clusterers. First, each of the cluster result computed by clusterer k is a partition $\prod_k$, the symbol $\prod_k=\{\pi_1,\pi_2,\ldots\pi_H\}$,$\pi_1 \cup \pi_2 \cup \ldots \cup \pi_H=$***Set***. Here π_i represents a certain cluster of the result of clusterer k. Then employ Kronecker Delta Function as the consensus function to set the value of CoA. The following is the Kronecker Delta Function: Here, $\pi_i(x)$ equals to 1 means tag x is in cluster π_i, otherwise it equals to 0, otherwise, it equals to 0.

$$s(x,y)=\sum_{i=1}^{H}\delta(\pi_i(x),\pi_i(y)),\delta(a,b)=\begin{cases}1, a \text{ and } b \text{ are both } 1\\ 0, \text{otherwise}\end{cases} \quad (1)$$

The main properties of the CoA matrix are as follows:

Property 1: Each cell of CoA represents how many clusterers put the two tags into the same cluster. Its value would not exceed the number of clusterers.

Property 2: The matrix is diagonally symmsetric. That means, for all cells, CoA(i, j) always equals to CoA(j, i).

Property 3: The graph represent the matrix may not be a connected graph.

Proof. There exists possibility that these tags belong to different clusters and all the clusterers put these tags into the different clusters. (See Example 2).

Example 2. There are 3 tags: '*fish*', '*sea*' and '*Philosophy*'. Three clusterers may put '*fish*' and '*sea*' into one cluster and put '*Philosophy*' into another, thus the CoA is not a connected. It contains two disconnected sub-graph. (Table 3 and Fig. 3)

Table 3. Cluster result of Example 2

Clusterer	Result
Clusterer 1	{*fish*, *sea*}, {*Philosophy*}
Clusterer 2	{*sea*, *fish*}, {*Philosophy*}
Clusterer 3	{*Philosophy*}, {*fish*, *sea*}

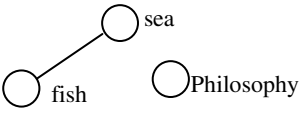

Fig. 3. Graph of Example 2

3.3 Decompose Graph

The graph can be decomposed into any number of sub-parts. It's a re-clustering based on the result of the first time clustering. The matrix represents a graph. We modify MST (Minimum Spanning Tree) algorithm to minimum spanning forest, and then decompose the forest to any number of parts according to the edge value of pairs of nodes. The pseudo code is as follows:

Algorithm 1: Decompose.

Input: N, number of part; G, graph
1. Let n = current partition of G and CL← Ø
2. while n < N
3. find the node with the smallest weight and split the edge.
4. calculate the distance between each partition and the tag.
5. for each partition
6. create cluster according to partition
7. add cluster to CL
8. Sort CL according to distance

Note that: (a) Each node is used to record the tag itself and its parent, if it is the root of a tree, the parent is NULL, otherwise it point to another tag. (b) The graph of the matrix may not be a connected graph. (c) In Step 3–4, while the number of parts is less than expectation, the loop continually split the forest.
The whole process of the algorithm is as follows:

Algorithm 2: Clustering Ensemble.

Input: n, clusterer size; ***Set***, similarity set
1. Use n clusterers to cluster the tags according to ***Set***
2. Generate CoA matrix G for $G(x, y) = \sum_{i=1}^{|C|} \delta(\pi_i(x), \pi_i(y))$
3. for each node u ∈G
4. do key[u] ←0 and parent[u] = NULL
5. key[r] ← 0
6. put all nodes into a priority queue Q

```
7.  while Q ≠Ø
8.    do u = extract-min(Q)
9.      for each v is adjacent to u
10.         do if v ∈Q and w(u, v) > key[v]
11.            then parent[v] = u
12.                 key[v] = w(u, v)
13.  return Decompose(parent, N)
```

Note that: (a) In Step 1, the clusterers are used to cluster the tags. (b) CoA is created in Step 2. (c) Maximum weight among all the neighbors and the parent of node v is recorded in key[v] and parent[v] respectively through Step 3 - 12.

4 SkyTagClustering Algorithm

Algorithm 2 is good in stability, but it is still with the following deficiencies:

- The user must pre-assign a parameter to set the number of clusters. The ideal situation is to let the algorithm itself find the proper number of clusters according to the data rather than manual set beforehand.
- The quality and efficiency of the clustering results depend on the basic algorithm chosen. If the basic algorithm is not good, the performance of CETClustering would be affected.

To solve the problems, we propose new algorithm based on skyline borrowing some ideas from [15]. As the authors mentioned in [15], under relatively lower dimension ($d < 12$), the two-scan algorithm version (TSA) has better performance, so our method employs it as the basic algorithm. The definition of k-dominate and the pseudo code of TSA can be referred in [15]. The algorithm asks for three parameters D, S and k, respective means the dataset, in our algorithm it is ***Set***, the data space and the parameter k for k-dominate.
There are several advantages for us to employ skyline as the basic algorithm:

a) Existing algorithm for skyline are relative efficient.
b) The performance of skyline is not affected by weight of dimension
c) Users do not need to control the number of clusters.

The steps of the SkyTagClustering algorithm are as follows: (1) Transform ***Set*** into proper form. (2) Iteratively extract skylines using basic skyline operator algorithm.

4.1 Data Transformation

We use the same preprocess method stated in Section 3 for SkyTagClustering. The similarity set ***Set*** should be transformed into proper form to fulfill the need of skyline operator.

In original ***Set***, 1.0 means the most correlated and 0.0 means the least correlated. The transformation is easy. In new ***Set***, we simply set the values as 1 minus the original value. Thus all the values denote the distance from a tag to core-tag. After transformation, the smaller the value, the closer it should be.

4.2 Iterative Extract Skylines

TSA is iteratively used to retrieve one skyline per time. Each time the skyline is found, the points in the skyline are removed from similarity set and used to create a cluster and the remaining data are reused to find new skyline until no point left (See Fig. 4).
There are some properties of the algorithm SkyTagClustering:

Property 1: The earlier found skyline is closer to the target tag.

Proof. For any tag T_n not in the skyline must be dominated by at least one tag T_s in skyline. This means there exist at least one tag in skyline more close to T_c than T_n.

Property 2: The algorithm is progressive. Thus it can quickly return the first cluster (the closest group of tags).

Proof. The skylines are iteratively computed. If only the first cluster is needed, the algorithm can return it and stop quickly.

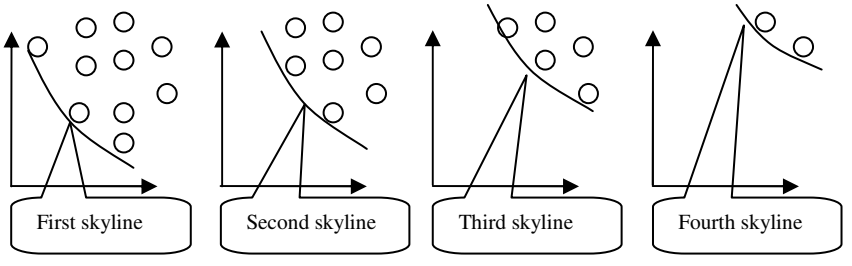

Fig. 4. Iteratively extract skylines

The pseudo code of this algorithm stated as follows:

Algorithm 3: k-dominant Skyline Clustering Algorithm(D, S, k)

```
1.  if k > D. dimension
2.     report error, k is set too large
3.  Transform the similarity set
4.  initialize set of cluster C = Ø
5.  while D != Ø
6.     cluster Ci = TSA(D, S, k)
7.     if |cluster| = 0 then report error, k is set too small
8.     remove the points in Ci from D
9.     add Ci to C
10. end while
11. return C
```

Note that: (a) Step 1 guarantees k > D. (b) Step 3 transforms similarity set to meet the requirement of skyline operator. (c) The skylines are iteratively computed by the help of TSA, for each skyline, a new cluster is created to put the points in. After this algorithm, a set of clusters C is formed.

5 Experiment and Performance Study

5.1 Experiment Setting

The goal of experiment is to find a good clustering algorithm for core-tag clustering. A modified K-means is applied as baseline method and we compare it with CETClustering (CE for short) and SkyTagClustering (SC for short) both on the aspects of efficiency and the quality.

Experiment Data: All the dataset are real dataset downloaded from http://del.icio.us from Nov 20 to Dec 15, 2007. There are a total of 234023 unique tags and 749971 unique web pages. All the data are processed by previous work mentioned in [4] so the processed data will be directly used.

Platform: All the experiments are implemented by java and are conducted on a PC with Intel Core2 Due CPU with 2G memory, running on Windows server 2003.

5.2 Efficiency Comparison

The efficiency of the three algorithms can be seen in Fig. 5 and 6. We set k (Size of candidate related tags) as 20 and 30 respectively, numbers of clusterers in CE as 3 (CE3) and 5 (CE5) respectively, and set the clustering tasks (number of core-tags) through 10 to 100 to test their speed. It is clear that SC has the best performance. CE

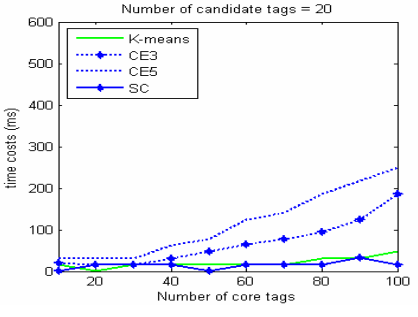

Fig. 5. Number of candidate tags = 20

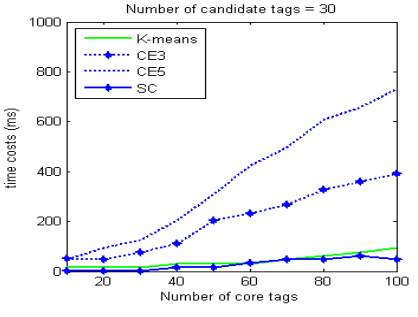

Fig. 6. Number of candidate tags = 30

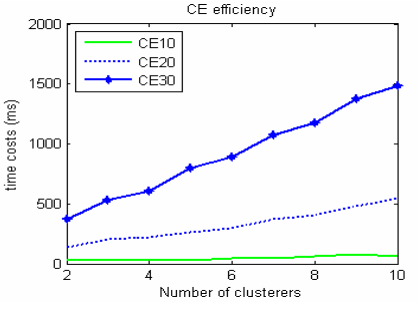

Fig. 7. CETClustering efficiency

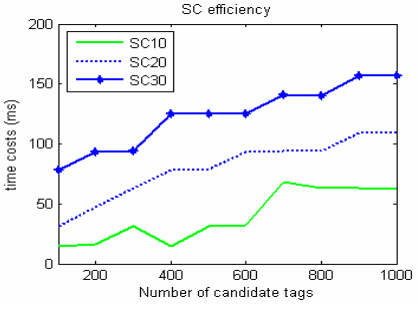

Fig. 8. SkyTagClustering efficiency

cost more time than the other two algorithms. It's quite reasonable. It costs at least five times longer than K-means version since it employs five K-means and does some extra work to get the final result. With increase number of K-means, time costs will increase undoubtedly.

Fig. 7 shows how clusterers size affects the efficiency of CE algorithm. We test the time costs by changing the number of potentially related tags for each core tag from 10 to 30 (CE10, CE20 and CE30). It is obvious that the time cost of CE is not linear increasing as the increasing of the number of clusterers, it's because besides the independent clustering of the clusterers this algorithm also does some work to combine the results of each clusterer to get final result, it would cost extra time. In efficiency experiment, CE performs worst and SC performs best.

Fig. 8 shows k affects the efficiency of SC algorithm. This algorithm is also tested by changing parameter k for each core-tag from 10 to 30 (SC10, SC20 and SC30). It can be seen that the number of candidate tags linearly affected the time costs of SC. Moreover, it is obvious that SC is a scalable algorithm, the clustering tasks increase from 100 -1000 but the time cost just doubled.

Above all, CETClustering algorithm doesn't perform well in the efficiency experiment because it employs several K-means. SkyTagClustering, on the contrary, performs very well both on the comparative and scalable test.

5.3 Remarks Clustering Result

As there is no authoritative benchmark method for research of web2.0, we ask three human to evaluate the quality of these algorithm versions: K-means, CE 3, CE 5 and SkyTagClustering. For each algorithm, there are 50 core-tags sample for evaluators to mark. For each core tag, each cluster of the result is assigned an index, the smaller the index the closer the cluster to the core tag. The evaluator should give a score according to the core tag and a set of clusters. The meaning of score is as follows:

S
- 5: Perfect. Tags are put into right cluster and the number of cluster is proper.
- 4: Very good. Most of the tags are right placed.
- 3: Good. A large part of the tags are right placed.
- 2: Fair. Some of the tags may not be put into right cluster.
- 1: Bad. Most of the tags are wrongly placed.

The method to evaluate the algorithm is very simple.

$$S_a = (\sum_{i=1}^{U} \sum_{j \in a} score_{aj}) / N_a \tag{4}$$

Here, S_a means final score of algorithm a, U is the number of evaluators, $score_{aj}$ is the score for each core tag get by algorithm a, N_a is the total number of scores of a. Table 5 shows the evaluate result of the four algorithms.

Table 4 indicates that SkyTagClustering has the best quality for it owns several advantages that other algorithms lack. CETClustering with 5 K-means ranked the second and 3 K-means ranked the third for they are more robust than the single K-means

Table 4. The scores of algorithms

Algorithm	Average Scores	Standard Deviation	Rank
K-means	2.71	0.51	4
CETClustering 3	2.96	0.52	3
CETClustering 5	3.19	0.33	2
SkyTagClustering	3.48	0.61	1

version. They can reduce the randomness of single K-means, the more number of clusterer, the more stable it is. However, as the increase of the cluterers, its efficiency would be affected. As the scores of CE3 and CE5 are very near but CE3 is more efficient than CE5, combining the efficiency and quality, CE3 outgoes CE5.

6 Conclusion and Future Work

We proposed two algorithms to handle the core-tag clustering task by using multi-similarity measurements. In conclusion, SkyTagClustering is considered to be the best algorithm for it has the best performance both on efficiency (10% - 50% faster than K-means) and qualification. CETClustering has a good performance in qualification. If it is used clustering algorithm faster than K-means, it would have better speed.

The future work includes: Employ some other clustering algorithm as the basic algorithm of CETClustering to do the clustering work and evaluate its performance. Find a more efficient skyline algorithm as the basic algorithm and let the algorithm create more proper number of clusters. For the whole project, we have done the first 6 steps. Multi-tags correlation problem and tag networks are still left. Our next work is to finish these two tasks based on previous work.

References

1. Mates, A.: Folksonomies – Cooperative Classification and Communication through Shared Metadata. In: Computer Mediated Communication, LIS590CMC (2004)
2. Hammond, T., Hannay, T., Lund, B., Scott, J.: Social Bookmarking Tools:A General Review. D-Lib Magazine (2005)
3. Hotho, A., Jäschke, R., Schmitz, C., Stumme, G.: Information retrieval in folksonomies: Search and ranking. In: Sure, Y., Domingue, J. (eds.) ESWC 2006. LNCS (LNAI), vol. 4011, pp. 411–426. Springer, Heidelberg (2006)
4. Xu, K., Chen, Y., Jiang, Y., Tang, R., Liu, Y., Gong, J.: A comparative study of correlation measurements for searching similar tags. In: Tang, C., Ling, C.X., Zhou, X., Cercone, N.J., Li, X. (eds.) ADMA 2008. LNCS, vol. 5139, pp. 709–716. Springer, Heidelberg (2008)
5. Fred, A., Jain, A.K.: Evidence Accumulation Clustering based on the K-means Algorithm. In: Proceedings of the Joint IAPR International Workshop (2002)
6. Voorhees, E., Gupta, N.K., Johnson-Laird, B.: The Collection Fusion Problem. In: The Third Retrieval Conference (1995)
7. Dietterich, T.G.: Ensemble methods in machine learning. In: Kittler, J., Roli, F. (eds.) MCS 2000. LNCS, vol. 1857, p. 1. Springer, Heidelberg (2000)

8. Quinlan, J.R.: Bagging, boosting, and C4.5. In: Proc. of the13th AAAI Conference on Artificial Intelligence. AAAI Press, Menlo Park (1996)
9. Oza, N.C.: Ensemble Data Mining Methods. NASA Ame Research Center (2000)
10. Strehl, A., Ghosh, J.: Cluster Ensembles – A Knowledge Reuse Framework for Combining Partitionings. AAAI, Menlo Park (2002)
11. Topchy, A., Jain, A.K., Punch, W.: Combining Multiple Weak Clusterings. In: ICDM (2003)
12. Borzsony, S., Kossmann, D., Stocker, K.: The Skyline Operator. In: ICDE (2001)
13. Tan, K.L., Eng, P.K., Ooi, B.C.: Efficient progressive skyline computation. In: VLDB (2001)
14. Chan, C.-Y., Jagadish, H.V., Tan, K.-L., Tung, A.K.H., Zhang, Z.: On high dimensional skylines. In: Ioannidis, Y., Scholl, M.H., Schmidt, J.W., Matthes, F., Hatzopoulos, M., Böhm, K., Kemper, A., Grust, T., Böhm, C. (eds.) EDBT 2006. LNCS, vol. 3896, pp. 478–495. Springer, Heidelberg (2006)
15. Chan, C.Y., Jagadish, H.V., Tun, K.L., Tung, A.K.H., Zhang, Z.: Finding k-Dominant Skylines in High Dimensional Space. In: SIGMOD (2006)
16. Papadias, D., Tao, Y.: An optimal and progressive algorithm for skyline. In: SIGMOD (2003)
17. Kossmann, K., Ramsak, F., Rost, S.: Shooting Stars in the Sky-An Online Algorithm for Skyline Queries. In: VLDB (2002)