Towards Enhancing Historical Analogy: Clustering Users Having Different Aspects of Events

Ryohei Ikejiri[1], Ryo Yoshikawa[2], and Yasunobu Sumikawa[3]

[1] Interfaculty Initiative in Information Studies, The University of Tokyo, Japan
[2] Department of Information and Media Studies, Nagoya Bunri University, Japan
[3] University Education Center, Tokyo Metropolitan University, Japan
ikejiri@iii.u-tokyo.ac.jp, yoshikawa.ryo@nagoya-bunri.ac.jp,
sumikawa-yasunobu@tmu.ac.jp

Abstract. Studying history can provide numerous benefits for finding meaningful connections or analogies over time. Several researchers have studied how to support promoting historical analogy by computer supported learning; however, supporting group discussions to promote the analogy still remains unexplored. In this paper, we propose a novel clustering algorithm to form users who have different aspects of the same past event in order to ease exchange ideas of what aspects they focus to analyze the event. We implemented our algorithms and evaluated them in terms of getting accuracies of forming users having different aspects. Experimental results proved that only our algorithm creates suitable groups.

Keywords: Clustering, analogy, collaborative learning, history education

1 Introduction

Studying and analyzing history-related data can provide numerous benefits including improved comprehension of the past and support for finding meaningful connections or analogies over time. Indeed, a researcher claims that history provides not only information on the past but also alternative solutions to similar modern issues [4]. One of the common goals of teaching and spreading the knowledge of history is to allow studying how people in the past tried to solve issues and problems, and then apply the acquired knowledge for proposing creative solutions to present issues [19]. Especially, teaching guidelines for high school education published by the Japanese government considers fostering the ability to apply historical knowledge to modern issues to be the advanced goal [6]. To support the modern history learning, researchers have developed effective learning methods [2, 16], algorithms mining past events similar to a given present event [25], and an interactive system that is useful in class [15].

For promoting historical analogy, finding similar past and present events plays a key role. However, how each person feels a similarity between past and present events is up to the person [12]. Furthermore, Fischer found that historical analogy often causes the misuse of analogy; in other words, we should do careful discussions in the case of using historical analogy [8]. According to [16], group learning in history with their discussions is effective to check the validity of each historical analogy. This study founds

a potential of the group learning; however, there is no algorithmic method to create groups consisted of persons.

Contributions. In this paper, we propose a novel clustering algorithm to promote historical analogy through group discussions. Our algorithm has two objectives to create groups: 1) finding two users, we call them a pair, who focus on the same aspects of an event and 2) aggregating two pairs, we call the aggregated two pairs a group, that have different aspects in the same event. Through these two steps, users can confirm correctness of their ideas within own pairs and can exchange them with members in the other pair. Compared to past works of clustering, the key contribution of this algorithm is to combine *not similar* data into a group as performed in the second step. Traditional clustering algorithms basically make groups by similar data such as finding similar documents to organize them [1], pixels to identify any parts of an image like face and landscape [9].

Example. We present examples of how to use our algorithm in a real situation. Recently, history teachers working in Japanese high school are required to have collaborative learning in class. If they can make good groups, students can have several positive effects, for example, from their discussions about connectivity between past and present events like Industrial Revolution and IT Revolution, they make their own knowledge deeper by explaining their ideas, know what other persons have ideas for the same learned things, find common things between the two events, think analogy between past and present, and so on. However, making groups for stimulating good collaborative learning is challenging; it is required to predict how well every student can get good outcomes. Our algorithm is useful to perform combining students who have same and different ideas about same things.

The contributions of this study are summarized as follows:

1. We present a novel clustering algorithm for creating a new environment for modern learning history.
2. Our clustering algorithm makes groups by combining not only similar users but also not similar pairs for promoting historical analogy through group discussions.
3. We evaluate that our algorithm creates groups composed of different aspects for the same event.

The remainder of this paper is organized as follows: Section 2 provides definitions we tackle in this paper. Section 3 summaries several related works. Section 4 explains our data collections. Section 5 describes how to create clusters. Section 6 shows experimental results, and Section 7 contains our conclusions.

2 Problem Definitions

Assumption. We assume that every user selects several event categories for a same event. Each user can select the one or more categories by what he/she focuses on analyzing aspects of the event. In this paper, we regard the selected event categories as aspects of the event and use them to create feature vectors as described in Section 5.2.

Input & Output. Let C be a set of event categories, C' be a power set of C. Given selected event categories by users C'', our algorithm outputs clusters of users Cu. Event

categories are usually predicted from given feature vectors. However, we treat them to create clusters; we first define feature vectors from the given event categories (C'') and make groups from the vectors.

3 Related Works

3.1 History Education

Drie and Boxtel reviewed previous studies on historical reasoning [5] and found six components of historical reasoning: asking historical questions, using sources, contextualization, argumentation, using substantive concepts, and using meta-concepts. However, previous studies did not take into account the students' ability to use historical causations as a means of solving present and future problems.

Mansilla examined in depth how students apply history to current problems successfully [2]. She used the Rwandan Genocide of 1994 as an example of a modern social problem and the Holocaust as a similar event in the past. As a result of her research, students were able to successfully use history to build an informed comparison base between both cases of genocide, recognize historical differences between them, appropriately apply historical modes of thinking to examine the genocide in Rwanda, and generate new questions and hypotheses about the genocide. However, Mansilla did not investigate what themes could enable other historical causations to connect with other modern problems.

Lee insists that an *usable historical framework* can make connections between events in the past and potentially between events in the past and present [19]. This framework must be an overview of long-term patterns of change, and be an open structure, capable of being modified, tested, improved, and even abandoned. This would enable students to assimilate new history to the existing framework. In addition, its subject should be human history, not some privileged sub-set of it. This is the underlying concept of a theoretical framework for using historical causations as a means of solving present and future problems. However, Lee did not explain how to create concrete themes for the usable historical framework and learning environment.

Ikejiri designed a competitive card game for high school students studying world history, where players can construct causal relations in the modern age by using historical causal relations [13]. This educational material includes a staged learning method for identifying causal relationships within modern societal problems by using references to historical causal relations. The effectiveness of this game was tested, and the results proved that this educational tool is effective in improving both the ability to associate past and current events with similar characteristics and to analyze causal relations in modern problems by referencing historical problems. Further, Ikejiri et al. designed another competitive card game that high school students can use to apply learning from the policies created in world history to create new policies that would revitalize Japan's economy [16]. The effectiveness of this game was tested, and it revealed that the number of policies proposed to invigorate Japan's economy increased after the high school students' use of this card game. These two studies provide effective methods for high school students to use history to analyze causal relations in modern problems by ref-

erencing historical problems and to cultivate alternative solutions to confront contemporary social problems. However, the theme that can connect historical contexts with modern society is set by a researcher each time manually. Thus, these learning methods are not automated according to modern context.

3.2 Clustering

Clustering is an important and fundamental technique in NLP, ML, and other computer science related research fields. Indeed, there are lots of survey papers such as comparing algorithms in theory and/or practical views [7, 27, 26] and domain-specific comparisons like text clustering [1], image processing [9]. We summarize the closest algorithms with their categories in this paper. More detailed categorizations are presented in the above survey papers.

Partitioning-based algorithm. We fist summarize one of the most basic and popular algorithms in many research fields. The intuitive idea of this kind algorithm is to divide data into groups satisfying 1) each group must contain at least one object, and (2) each object must belong to exactly one group. One of the most famous algorithms of is K-means [20] that updates the centers of clusters by iteratively computing averages of all points and coordinates representing the arithmetic mean. This process continues until some criteria are satisfied. In addition to K-means, PAM [18], CLARA [17], and CLARANS [21] are also famous and used widely.

Hierarchy-based algorithm. For many data, we can naturally define hierarchical relationship among data and can represent them as a dendrogram whose leaves represent the data. To construct the hierarchical clusters, there are two kinds of algorithms: agglomerative (bottom-up) or divisive (top-down) ones. The former algorithm creates clusters for each data and recursively merges them. The latter one starts with the whole dataset and recursively dived them into sub-clusters. These processes are basically performed until some criteria are satisfied, such as k clusters can be created. As for the hierarchy-based algorithms, BIRCH [29], CURE [10], and ROCK [11] are proposed.

Distribution-based algorithm. This kind of algorithms assumes that each data in the same cluster is generated from the same distribution. Gaussian mixture model (GMM) [23] is one of the most popular algorithms of this kind category. GMM considers that every data is generated from several Gaussian distributions. DBCLASD [28] is another popular algorithm of the distribution-based algorithm. This is a dynamic incremental algorithm considering nearest neighborhood.

Graph-based algorithm. Once we regard each data is a node, we can construct a graph by representing an edge whose weight represents a score of similarity between two data. Spectral clustering [24] is a typical algorithm performing clustering on a graph.

Our algorithm is designed as 2 steps of partitioning-based approach; however, we compare our algorithm with the 4 kinds of algorithms in Sec. 6. This is because it is possible to regard the algorithm as a hierarchy- or distribution-based ones as we perform grouping twice and generate the groups from different distributions at each step. As a graph-based algorithm is another view of clustering, we use spectral clustering as a baseline. In Sec. 6, we describe that no baselines can be fit to the purpose of this study.

Table 1. Example events. The abbreviated names of categories are used: Reign (Rg), Diplomacy (Dp), War (Wr), Production (Pr), Commerce (Cr), Study (St), Religion (Rl), Literature and Thought (LT), Technology (Tc), Popular Movement (PM), Community (Cn), Disparity (Ds) and Environment (En)

Event	Categories
Agnes Chan named UNICEF Regional Ambassador for East Asia and Pacific Region.	Dp, Cn and TL
The World Strikes a Deal on Climate Change.	En
Paris attacks.	Dp, Rg and PM
ISIS Terrorists Strike on Three Continents.	Dp, Rl, Wr and PM
Same-Sex Marriage Debate.	TL and Cn
Ebola outbreak.	En, St and Tc
The Scottish independence referendum.	Rg, PM and Cn

4 Data Collection

4.1 Event Classes

In this paper, we use 13 categories: Reign (Rg), Diplomacy (Dp), War (Wr), Production (Pr), Commerce (Cr), Study (St), Religion (Rl), Literature and Thought (LT), Technology (Tc), Popular Movement (PM), Community (Cn), Disparity (Ds) and Environment (En). They were described in [14, 15] as a proposal of an event category list to define curriculum of teaching history with connecting past and present. These categories are based on definitions of Encyclopedia of Historiography [22]. We show example events for the 13 categories in Tab. 1.

4.2 Present Event

We assume that every user reads news papers or any other kinds of articles like Wikipedia describing present social issues. In general, an event can be classified into several classes. For example, if we read the Wikipedia article[1] to know what the 2014 West Africa Ebola outbreak caused in our life, we can see that it killed many both human and nonhuman (environment event), we developed a vaccine (technology event), some researchers report the details and their statistics (study event), and so on.

Similar to Tab. 1, users reading the contents of an article of present event select more than one category for each event by what they want to focus on.

5 Methodology

5.1 Overview

As our purpose is to stimulate users exchanging their ideas and discussing them, we design the algorithm with the following requirements.

[1] https://en.wikipedia.org/wiki/West_African_Ebola_virus_epidemic

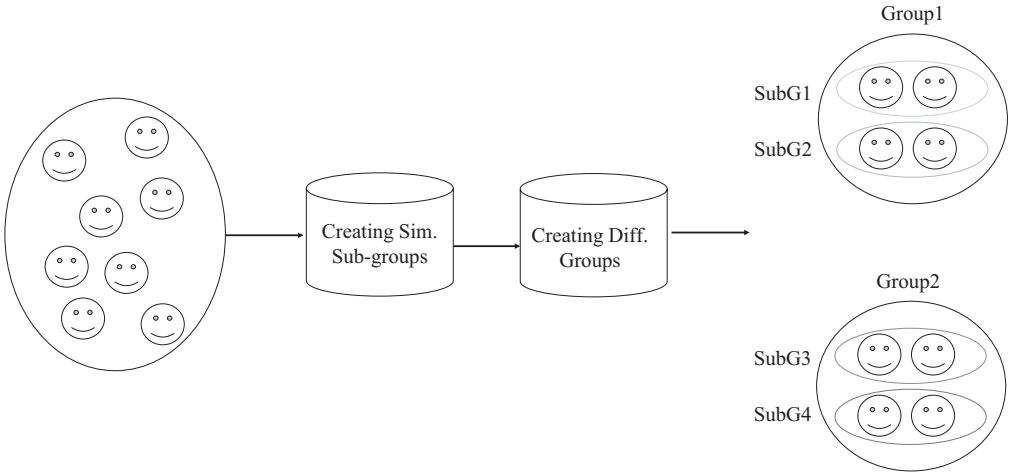

Fig. 1. Overview of our algorithm.

1. Users in each group have different aspects for the same event.
2. There are at least two users who have the same aspect in a group.

The first requirement is the key idea of our algorithm. As described in Sec. 1, grouping users who have different ideas is an effective way to stimulate their discussions. The last one is helpful to discuss in case that one user misses some important ideas.

Fig. 1 shows an overview of our algorithm. Our algorithm first takes information about what aspects each user focus on for an event. Then, to satisfy the second requirement, it forms two users according to their selected event categories. We call the two users a pair. Last, our algorithm combines two pairs for the first requirement.

We describe each step in the remainder of this section.

5.2 Feature Vector Creation

We first take event categories selected for an event by users. We then convert the categories to a feature vector whose elements are represented by 0 or 1. Formally, we create a feature vector for ith user as follows:

$$f_{i_k} = \delta(c_k, C'), c_k \in C, 1 \leq k \leq \mid C \mid \tag{1}$$

where C is a set of all event categories, C' is a set of event categories selected by a user, and the function δ returns 1 if the first argument is included in the second argument; otherwise, it returns 0. This equation defines the kth element of the feature vector.

5.3 Combining Similar Two Data

After converting the categories into feature vectors, we create pairs of users by their similarities. We measure the similarity by counting how many common categories are

selected; the more the number of common categories uses select, the more they are similar to each other.

The formal definition of measuring pair similarity is given as follows:

$$PairSim(f_i, f_j) = \sum_{k}^{|C|} And(f_{i_k}, f_{j_k}) \tag{2}$$

where f_i and f_j represent feature vectors for ith and jth users and the function *And* applies the AND logical operator to the two arguments. As described in the previous section, each element is a binary; therefore, sum of the results of *And* represents how many common categories are selected by two users.

We then solve the following maximum problem.

$$\begin{aligned} max \quad & PairSim(\boldsymbol{P}) \\ s.t. \quad & \bigcap P_i = \emptyset \\ & |P_i| \geq 2 \end{aligned}$$

where $\boldsymbol{P}$ is a set of pairs, and $P_i \in \boldsymbol{P}$ is a set of users.

Algorithm 1 shows an algorithm solving the above minimum problem. This problem is essentially *Knapsack problem*; we have several knapsacks, and then determine the combinations of two feature vectors so that each knapsack can include at most two users and the total variance of the pairs is as large as possible. It is well known that the its optimization problem is NP-hard; there is no known polynomial algorithm providing a solution. To solve this problem, we traverse a tree representing candidates of pairs and then determine the best pairs by calculating scores of all candidates.

Algorithm 1 Algorithm of combining two similar data

Input: A set of feature vectors F
Output: A set of pair P

```
1: Function CreatePairs(F)
2: P = ∅
3: WorkList = F
4: while |WorkList| > 0 do
5:     (s1, s2) = FindBestPair(WorkList)
6:     P.add((s1, s2))
7:     WorkList.remove(s1)
8:     WorkList.remove(s2)
9: end while
10: return P
```

5.4 Creating Groups

After creating the pairs, we combine two pairs by connecting two pairs that focus on *different* aspects of an event. In other words, we combine two pairs whose common

Algorithm 2 Algorithm of finding the best pair of given list

Input: A set of feature vectors *WorkList*
Output: A pair of two users *BestPair*

```
1: Function FindBestPair(WorkList)
2: val = 0
3: BestPair = None
4: for i = 0; i < WorkList.size(); i + + do
5:     s1 = WorkList[i]
6:     for j = i + 1; j < WorkList.size(); j + + do
7:         s2 = WorkList[j]
8:         sval = PairSim(s1, s2)
9:         if val < sval then
10:            val = sval
11:            BestPair = (s1, s2)
12:        end if
13:    end for
14: end for
15: return BestPair
```

event categories is as less as possible. As we consider pair-level selected event categories, we first define a feature vector for a pair. Formally, we define a feature vector for a pair (f_i, f_j) as follows:

$$p(f_i, f_j)_k = f_{i_k} + f_{j_k}, 1 \leq k \leq |C| \tag{3}$$

Then, We define the scores of similarity between pairs as follows:

$$GroupSim(p_i, p_j) = \sum_{k}^{|C|} p_{i_k} * p_{j_k} \tag{4}$$

where p_i and p_j represent feature vectors for ith and jth pairs.

We then solve the following minimum problem.

$$\begin{aligned} min \quad & GroupSim(\boldsymbol{G}) \\ s.t. \quad & \bigcap G_i = \emptyset \\ & |G_i| \geq 2 \end{aligned}$$

where $\boldsymbol{G}$ is a set of groups, and $G_i \in \boldsymbol{G}$ is a set of pairs.

Algorithm 3 shows an algorithm solving the above maximum problem. Similar to Algorithm 1, this algorithm solves Knapsack problem. We traverse a tree representing candidates of groups to compare scores of all candidates.

6 Experimental Evaluation

6.1 Setup

Data collection. We collect practical data that are event categories selected by 40 third-year high school students in Japanese public schools who learn world history for a real

Algorithm 3 Algorithm of creating groups by combining two pairs.

Input: A set of feature vectors for pair FP
Output: A set of group G

Function CreateGroups(FP)
$G = \emptyset$
$WorkList = FP$
while $|WorkList| > 0$ **do**
  $(p1, p2) = FindBestGroup(WorkList)$
  $G.add((p1, p2))$
  $WorkList.remove(p1)$
  $WorkList.remove(p2)$
end while
return G

Algorithm 4 Algorithm of finding the best groups of given list.

Input: A set of feature vectors *WorkList*
Output: A group of two pairs *BestGroup*

Function FindBestGroup(*WorkList*)
$val = 0$
$BestGroup = None$
for $i = 0; i < WorkList.size(); i++$ **do**
  $p1 = WorkList[i]$
  for $j = i + 1; j < WorkList.size(); j++$ **do**
    $p2 = WorkList[j]$
    $sval = GroupSim(p1, p2)$
    if $val < sval$ **then**
      val = sval
      $BestGroup = (p1, p2)$
    end if
  end for
end for
return *BestGroup*

Table 2. Statistics of data collection

Number of users	40
Number of present events	1

present event. Japanese labor problem is set as the present event in this case. We show the statistics we used in this paper in Tab. 2.

Baselines. We compare our algorithm with the following baselines.

– **k-means**: this is one of the most popular partitioning-based algorithms. This algorithm creates k clusters by assigning a cluster label for each point if it is closer to the cluster's centroid than any other centroid.

- **Birch**: this is a hierarchy-based algorithm. Our algorithm can be seen as an agglomerative hierarchy-based one as it performs creating groups twice; therefore, we compare our algorithm with this one.
- **GMM**: this algorithm is also widely used to create clusters as a distribution-based approach.
- **Spectral**: this algorithm is a graph-based approach, and is a commonly used if the structure of the individual clusters is highly non-convex.

Parameters. For our purpose, we assume that there are at least 2 users having the same aspects and 2 pairs having different aspects in each group. Therefore, we combine 4 users as a group. For this, we set k (number of clusters) as a result of dividing the number of uses by 4 in all algorithms.

Measurements. We evaluate all algorithms with three measurements as follows:

- **Size of clusters**: This measurement means that how many users are grouped for each cluster.
- **MinDist**: We measure the minimum Euclidean distance for each cluster.
- **Similarity of Inner-cluster**: We average all Euclidean distances for all combinations of feature vectors of a cluster.
- **User similarity**: This means that how similar two users in the same pair in average. We measure this score by Eq. (2).
- **Pair similarity**: This is an average Euclidean distance between two pairs in a group. Eq. (4) is used to measure this similarity.
- **Quality of clustering**: Calinski and Harabaz (CH) score [3] is one of the widely popular measuring qualities of clusters. This measurement is defined as follows:

$$CH(k) = \frac{(n-k)}{(k-1)} \times \frac{B(k)}{W(k)}$$

where $B(k)$ and $W(k)$ are sums of squares for the k clusters of inner- and inter-clusters, respectively. n is a number of clustered data. Intuitively, if all data in a cluster are close to each other, and data between different clusters are not close, we can say that the quality of the clusters is high. Thus, the higher the score is, the better quality of the cluster is. Even though the purpose of our algorithm includes aggregating not similar pairs, we measure the qualities of clusters to take more in-depth discussions.

6.2 Discussions of Results

At the beginning, we show how many users are grouped in Tab. 3. Looking at the first rows (Size) of all algorithms, kmeans, Birch, and GMM algorithms fail to include more than 2 users in a few clusters (C7 in GMM, C9 for Birch, and C10 for all) whereas our algorithm included 4 users in all clusters. Spectral outputs clusters having at least 2 users for each cluster; however, all users in 4 clusters (C2, C4, C5, and C6) selected the completely the same event categories. As our purpose is to promote group discussions and to combine users having different ideas, this result does not fit to the purpose.

Therefore, we can conclude that our algorithm is the best for the purpose of this study. In other words, if we create groups of users to promote historical analogy through their discussions, our algorithm is the best way compared with the 4 baselines.

Next, we analyze the qualities of all clusters. We first show minimum distances in each cluster created by all algorithms in Tab. 3. 4 baselines created 5 or 6 pairs whose users input completely same categories whereas our algorithm did 7 pairs. In contrast, looking at inner-cluster (sum of distances of all combinations in each cluster), there are large clusters (C3 for kmeans and spectral, C7 for Birch, and C1 and C2 for GMM) in baselines whereas our algorithm created almost same sizes for 8 clusters (from C1 to C8). Indeed, out algorithm is the smallest for scores of standard deviation for inner-cluster. We can find that our algorithm fails to combine different 2 pairs as a group for 2 clusters (C9 and C10). To increase the diversity of aspects, we will take variations of selected categories to group users as a future work.

We have checked qualities of each cluster in the above discussions. Next, we analyze the qualities of whole clusters in Tab. 4. First, we can see that our algorithm naturally achieved the lowest CH score as our algorithm combines not similar pairs at the second phase. This result indicates that our algorithm generates clusters whose sizes are relatively large. Indeed, in our algorithm, the average of minimum distances of inner-cluster (2nd column) is the smallest whereas the average total distances between data in each cluster is high following to Birch.

Figs. 2, 3, 4, 5, and 6 show how each algorithm assigns a cluster label to all user. As dimensional sizes of the feature vectors are 13, we decomposed them into 2-dimensional by singular-value decomposition (SVD). We can see that our algorithm tends to assign the same labels to close 2 users.

Table 3. Sizes, minimum distances in a cluster, and total distances between data in a cluster for each cluster. Each cluster is labeled by a unique name from C1 to C10. "-" means that there is no valid data to measure

		C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	Ave.	Std. Dev.
kmeans	Size	2	11	8	4	3	2	2	2	5	1	4	3.033
	MinDist	1.732	0.0	0.0	1.414	1.414	0.0	1.0	1.732	0.0	0.0	0.729	0.753
	Inner-cluster	1.732	0.0	35.113	11.295	5.382	0.0	1.0	1.732	5.656	-	6.878	10.554
Birch	Size	3	7	2	2	2	4	16	2	1	1	4	4.335
	MinDist	1.414	0.0	1.732	1.732	1.732	0.0	0.0	1.414	0.0	0.0	0.802	0.809
	Inner-cluster	4.878	21.999	1.732	1.732	1.732	7.292	68.709	1.414	0.0	0.0	10.948	20.233
GMM	Size	12	8	3	2	4	2	1	5	2	1	4	3.346
	MinDist	0.0	0.0	0.0	1.414	1.414	1.414	0.0	0.0	1.732	0.0	0.597	0.736
	Inner-cluster	19.052	35.431	2.828	1.414	11.799	1.414	-	5.656	1.732	-	9.915	11.280
Spectral	Size	3	11	6	4	3	2	3	2	3	3	4	2.569
	MinDist	1.414	0.0	1.414	0.0	0.0	0.0	0.0	1.0	1.732	1.732	0.729	0.753
	Inner-cluster	4.878	0.0	30.381	0.0	0.0	0.0	2.0	1.0	5.464	5.464	4.918	8.772
Proposed	Size	4	4	4	4	4	4	4	4	4	4	4	0.0
	MinDist	0.0	0.0	0.0	0.0	1.414	1.732	0.0	1.732	0.0	0.0	0.487	0.749
	Inner-cluster	11.948	11.922	9.797	11.922	11.032	13.512	11.103	12.385	0.0	0.0	9.362	*4.769*

Table 4. Results of all clusters.

Algorithm	Quality	Ave. MinDist	Inner-cluster
k-means	9.834	0.729	6.191
Birch	10.022	0.802	10.948
GMM	9.018	0.597	7.932
Spectral	9.714	0.729	4.918
Proposed	*1.740*	*0.487*	*9.362*

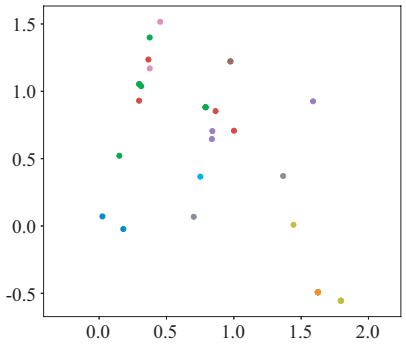

Fig. 2. Results of k-means.

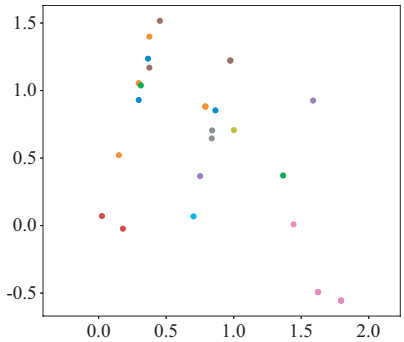

Fig. 3. Results of Birch.

Finally, we measure how well our algorithm creates groups by analyzing similarities of two users are in each pair and differences of two pairs are in each group, and show them in Tab. 5. We can see that all pairs tend to have similar interesting in aspects. In contrast, almost of distances between pairs are relatively longer than ones between pairs. Therefore, we can conclude that our algorithm successfully created groups.

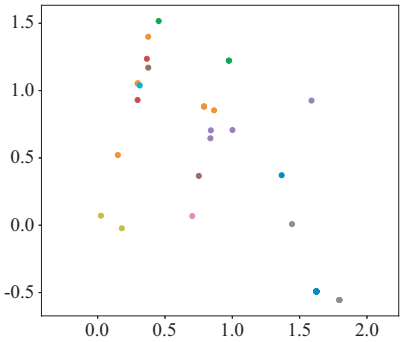

Fig. 4. Results of GMM.

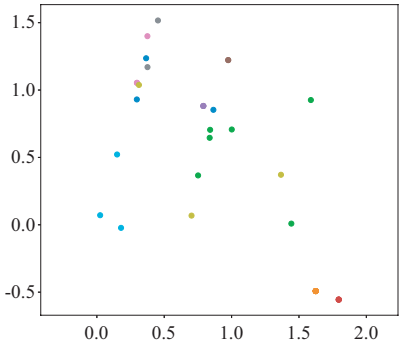

Fig. 5. Results of spectral.

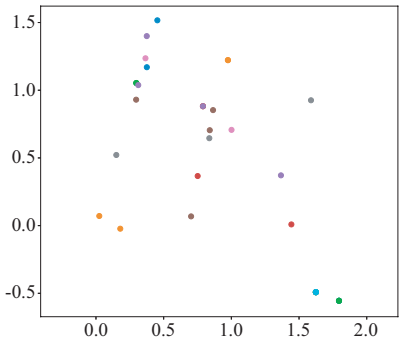

Fig. 6. Results of our algorithm.

7 Conclusions

Studying history provides numerous benefits including support for finding meaningful connections over time. In this paper, we introduce a novel clustering technique for

Table 5. User and pair similarities of our algorithm.

	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10
Dist(p1)	0.0	0.0	0.0	0.0	1.414	1.732	0.0	1.732	0.0	0.0
Dist(p2)	1.0	1.732	0.0	1.732	2.236	2.0	1.732	2.236	0.0	0.0
Dist(p1, p2)	2.692	2.397	2.449	2.397	1.322	2.061	2.179	1.581	0.0	0.0

creating new collaborative learning platform specialized for history. Our clustering algorithm has two phases: combining two users that have similar interesting in an event and combining two pairs that have different interesting in the event.

In the future, we plan to analyze *1) how well users can discuss with their own group members*. In addition, *2) we will propose more sophisticated grouping algorithm.* As we analyzed in our experimentations, our algorithm combines 4 users who have completely same aspects; it might fall into local optima. Finally, *3) we will analyze robustness of our algorithm.* In this paper, we used real data generated by Japanese high school students; however, we will check results in different datasets.

Acknowledgments

This work was supported by JSPS KAKENHI Grant Number 16K16314.

References

1. Aggarwal, C.C., Zhai, C.: A Survey of Text Clustering Algorithms, pp. 77–128. Springer US, Boston, MA (2012)
2. Boix-Mansilla, V.: Historical understanding: Beyond the past and into the present. Knowing, Teaching, and Learning History: National and International Perspectives pp. 390–418 (2000)
3. Caliński, T., Harabasz, J.: A dendrite method for cluster analysis. Communications in Statistics-Simulation and Computation 3(1), 1–27 (1974)
4. David, J.S.: A History of the Future, vol. 41. History and Theory, Theme Issue (2002)
5. van Drie, J., van Boxtel, C.: Historical reasoning: Towards a framework for analyzing students' reasoning about the past. Educational Psychology Review 20(2), 87–110 (2008)
6. Ministry of Education, Culture, S.S., Technology: Japan course of study for senior high schools (in japanese) (2014), accessed 13 May 2020
7. Fahad, A., Alshatri, N., Tari, Z., Alamri, A., Khalil, I., Zomaya, A.Y., Foufou, S., Bouras, A.: A survey of clustering algorithms for big data: Taxonomy and empirical analysis. IEEE Transactions on Emerging Topics in Computing 2(3), 267–279 (Sept 2014)
8. Fischer, D.H.: Historians' Fallacies : Toward a Logic of Historical Thought. New York: Harper & Row, Publishers (1970)
9. Fu, K., Mui, J.: A survey on image segmentation. Pattern Recognition 13(1), 3–16 (1981)
10. Guha, S., Rastogi, R., Shim, K.: Cure: An efficient clustering algorithm for large databases. pp. 73–84. SIGMOD'98, ACM, New York, NY, USA (1998)
11. Guha, S., Rastogi, R., Shim, K.: Rock: a robust clustering algorithm for categorical attributes. pp. 512–521. ICDE'99 (March 1999)
12. Holyoak, K.J., Thagard, P.: Mental Leaps: Analogy in Creative Thought. MIT Press (1980)

13. Ikejiri, R.: Designing and evaluating the card game which fosters the ability to apply the historical causal relation to the modern problems. Japan Society for Educational Technology 34(4), 375–386 (april 2011), (in Japanese)
14. Ikejiri, R., Sumikawa, Y.: Developing a mining system to transfer historical causations to solving modern social issues. WHA'16 (2016)
15. Ikejiri, R., Sumikawa, Y.: Developing world history lessons to foster authentic social participation by searching for historical causation in relation to current issues dominating the news. Journal of Educational Research on Social Studies 84, 37–48 (2016), (in Japanese)
16. Ikejiri, R., Fujimoto, T., Tsubakimoto, M., Yamauchi, Y.: Designing and evaluating a card game to support high school students in applying their knowledge of world history to solve modern political issues. ICoME'12, Beijing Normal University (2012)
17. Kaufman, L., Rousseeuw, P.J.: Finding Groups in Data: An Introduction to Cluster Analysis. John Wiley (1990)
18. Kaufman, L., Rousseeuw, P.J.: Partitioning Around Medoids (Program PAM), pp. 68–125. Wiley-Blackwell (2008)
19. Lee, P.: Historical literacy: Theory and research. International Journal of Historical Learning, Teaching and Research 5(1), 25–40 (2005)
20. Macqueen, J.: Some methods for classification and analysis of multivariate observations. In: 5-th Berkeley Symposium on Mathematical Statistics and Probability. pp. 281–297 (1967)
21. Ng, R.T., Han, J.: Clarans: a method for clustering objects for spatial data mining. IEEE Transactions on Knowledge and Data Engineering 14(5), 1003–1016 (Sept 2002)
22. Ogata, I., Kato, T., Kabayama, K., Kawakita, M., Kishimoto, M., Kuroda, H., Sato, T., Minamizuka, S., Yamamoto, H.: Encyclopedia of historiography. koubundou (1994)
23. Rasmussen, C.E.: The infinite gaussian mixture model. pp. 554–560. NIPS'99, MIT Press, Cambridge, MA, USA (1999)
24. Shi, J., Malik, J.: Normalized cuts and image segmentation. Tech. rep. (2000)
25. Sumikawa, Y., Ikejiri, R.: Mining historical social issues. IDT'15, vol. 39, pp. 587–597. Springer International Publishing (2015)
26. Xu, D., Tian, Y.: A comprehensive survey of clustering algorithms. Annals of Data Science 2(2), 165–193 (Jun 2015)
27. Xu, R., Wunschm, D.: Survey of clustering algorithms. IEEE Transactions on Neural Networks 16(3), 645–678 (May 2005)
28. Xu, X., Ester, M., Kriegel, H.P., Sander, J.: A distribution-based clustering algorithm for mining in large spatial databases. pp. 324–331. ICDE'98, IEEE Computer Society, Washington, DC, USA (1998)
29. Zhang, T., Ramakrishnan, R., Livny, M.: Birch: An efficient data clustering method for very large databases. pp. 103–114. SIGMOD'96, ACM, New York, NY, USA (1996)