Information Systems 64 (2017) 1–11

Contents lists available at ScienceDirect

Information Systems

journal homepage: www.elsevier.com/locate/is

A K-partitioning algorithm for clustering large-scale spatio-textual data

Dong-Wan Choi[a], Chin-Wan Chung[b,c,*]

[a] School of Computing Science, Simon Fraser University, Burnaby, BC, Canada
[b] Chongqing Liangjiang KAIST International Program, Chongqing University of Technology, Chongqing, China
[c] School of Computing, Korea Advanced Institute of Science and Technology, Daejeon, Korea

A R T I C L E I N F O

Keywords:
Spatio-textual similarity
K-means clustering
K-medoids clustering
K-prototypes clustering
Expected distance
Grid partitioning

A B S T R A C T

The volume of spatio-textual data is drastically increasing in these days, and this makes more and more essential to process such a large-scale spatio-textual dataset. Even though numerous works have been studied for answering various kinds of spatio-textual queries, the analyzing method for spatio-textual data has rarely been considered so far. Motivated by this, this paper proposes a *k*-`means` based clustering algorithm specialized for a massive spatio-textual data. One of the strong points of the *k*-`means` algorithm lies in its efficiency and scalability, implying that it is appropriate for a large-scale data. However, it is challenging to apply the normal *k*-`means` algorithm to spatio-textual data, since each spatio-textual object has non-numeric attributes, that is, textual dimension, as well as numeric attributes, that is, spatial dimension. We address this problem by using the expected distance between a random pair of objects rather than constructing actual centroid of each cluster. Based on our experimental results, we show that the clustering quality of our algorithm is comparable to those of other *k-partitioning* algorithms that can process spatio-textual data, and its efficiency is superior to those competitors.

1. Introduction

Nowadays, spatio-textual objects are prevalent with the fact that user created contents are mainly uploaded and created by mobile subscribers. For example, in Twitter[1], average number of tweets sent per day is about 500 million, and the 76% of Twitter users are mobile subscribers. Similarly in Foursquare[2], the number of check-ins, each of which consist of a location and some textual descriptions, is known to be millions more every day. Thus, the volume of spatio-textual data is already huge, and even grows in an extremely fast manner.

With this background, there have been extensive works on processing spatio-textual data in the last several years [1–5]. Most of them have highly concentrated on "how to process spatio-textual queries", namely *spatial keyword search*. In other words, their goal is commonly to find the object(s) satisfying some spatial and textual conditions while trying to minimize the query processing cost. However, it is also important to analyze the spatio-textual data at a high-level, and unfortunately none of the works above are intended for data analysis.

In this paper, we study a problem of clustering spatio-textual data motivated by the fact that clustering is one of the most fundamental tool in data analysis. In particular, we focus on extending the *k*-`means` algorithm for a massive volume of spatio-textual dataset to be efficiently processed. *k*-`means` still remains one of the most popular data processing algorithm over half a century especially due to its simplicity and scalability [6]. To the best of our knowledge, this is the first work on applying the *k*-`means` clustering algorithm to spatio-textual data.

Not to mention that clustering itself is a central problem in computer science, spatio-textual clustering can play a key role in many practical applications as follows:

- **Social media data analysis**. With the rise of social media platforms such as Twitter and Facebook, the importance of extract-

* Corresponding author at: Chongqing Liangjiang KAIST International Program, Chongqing University of Technology, Chongqing, China.
E-mail addresses: dongwanc@sfu.ca (D.-W. Choi), chung_cw@kaist.ac.kr (C.-W. Chung).
[1] https://twitter.com
[2] https://foursquare.com

http://dx.doi.org/10.1016/j.is.2016.08.003
Received 10 December 2015; Received in revised form 24 June 2016; Accepted 8 August 2016
Available online 28 September 2016

ing interesting patterns from the large amount of social media data is getting more and more increasing. As mentioned earlier, a large portion of such social media data is generated by mobile users, and therefore has location information as well as textual information. Thus, by finding spatio-textual clusters, we can reveal some underlying characteristics of social media.

- **Data cleaning in location-based systems**. Spatio-textual objects are also generated from various location-based services such as Foursquare. In many cases, some duplicated objects can be created by different users even if those objects are semantically the same. For example, the dataset containing venue information in Foursquare might have “Domino Pizza”, “Domino's Pizza”, and “Pizza Store” in a neighborhood even though they actually represent the identical place. Spatio-textual clustering can help to eliminate such duplicate venues by merging venues in the same cluster into one representative.
- **Preprocessing for spatial keyword queries**. Spatio-textual clustering can also be utilized as a preprocessing phase in spatial keyword search. Many spatial keyword query processing algorithms exploit underlying preprocessed structure such as the R-tree based index. Basically, in such a preprocessed index, it is more beneficial to gather spatially and textually similar objects into the same entry. This can be guided by utilizing the preliminary spatio-textual cluster information.

There are a rich body of works on the *k*-`means` algorithm in the literature, which produce a variety of derivatives. Since the key process of *k*-`means` is to compute and update the mean value of each cluster, most *k*-`means` family algorithms assume that each data object only contains numeric attributes. This assumption makes a big challenge in applying the *k*-`means` algorithm to spatio-textual data as each object contains both numeric (spatial) and non-numeric (textual) attributes.

To address the challenge above, we first observe that it suffices to compute the *expected distance* between a random object in each cluster and the object under consideration rather than measuring the distance from the virtually constructed spatio-textual centroid of a cluster. By doing so, we can reduce the cost of computing pairwise textual distances. Furthermore, we devise an effective technique for initializing *k*-`means` for spatio-textual data, which is commonly the most important and challenging task for *k*-`means` derivatives to improve not only the quality of resulting clusters but also the efficiency.

Our main contribution is summarized as follows:

- We firstly investigate the problem of clustering a large scale spatio-textual data. Spatio-textual clustering has many real applications such as social media data analysis, location-based data cleaning, and preprocessing of spatial keyword querying.
- We propose a modified version of the *k*-`means` clustering algorithm specialized for spatio-textual data. Our solution utilizes the expected pairwise distance and includes an effective initialization technique.
- We provide extensive experimental results using various real datasets. The experimental results show that our proposed algorithm is not only fast enough to tackle a massive spatio-textual dataset, but also fairly effective compared with the existing clustering algorithms that can process spatio-textual data without any modification yet requires a long running time.

1.1. Organization

The paper is organized as follows: Section 2 describes the related work on spatio-textual clustering. Section 3 formally defines the problem of clustering spatio-textual data. Section 4 presents our *k*-`means` based clustering algorithm, which is experimentally evaluated in Section 5. Lastly, Section 6 concludes the paper with a concluding remark.

2. Related work

Spatio-textual similarity search. The goal of our problem is basically to find the groups of objects that are spatially and textually similar to each other. Therefore, one of the most related categories of existing works is the *spatio-textual similarity search*, which aims at finding the closest object to the query object in terms of both spatial distance and textual relevance. There are a number of works on this topic that have been proposed [1–5] in the last several years. It is far beyond of this paper to scrutinize all those works. One can refer to the general survey on the spatial keyword query, which is well organized by Cao et al. in [7].

As mentioned earlier, all these methods are focused on how to find one or *k* nearest objects in terms of spatial and textual aspects. This is inherently different from our main purpose aiming at finding *k* spatio-textual clusters even though we follow the general definition of the spatio-textual distance in these works.

Spatial clustering. Focusing on spatial data mining, many clustering algorithms have been developed and widely utilized [8–13,6,14]. In this type of algorithms, a data object is regarded as a point in a metric space, and thus it is assumed that each object consists of only numeric attributes. The most popular clustering algorithms in this category is undoubtedly *k*-`means` [8], which itself has produced extensive derivative algorithms. Among those numerous variants, the most commonly used one in practice is the Lloyd's version [9], where the initial *k* seed objects are just selected randomly. Some derivatives [13,6] attempt to overcome the disadvantages of the Lloyd's algorithm by carefully choosing the initial *k* objects. Apart from the *k*-`means` family algorithms, there is another kinds of algorithms following the density-based clustering scheme [11,12,14], which is proposed to effectively capture the clusters of arbitrary shapes. All the algorithms above only allow the numeric data, and hence are not directly applicable to our spatio-textual data.

Another important branch of *k-partitioning* algorithms is *k*-`medoids` [15,16]. In *k*-`medoids` family algorithms, rather than maintaining *k* centroids, we determine *k* representative objects, called *medoids*, among the given dataset. One advantage of *k*-`medoids` is that it can be applied to non-numerical data as well as numerical data since it suffices to define a distance measure between any two objects regardless of their attribute types. However, *k*-`medoids` and its derivatives suffer from the high time complexity, which makes it difficult to process a massive spatio-textual data we are focusing on.

Categorical clustering. Speaking of only the textual aspect of our problem, it can fall in the literature of the categorical clustering methods in the sense that the textual information is represented as a set of keywords and this can be seen as a categorical attributes. Many algorithms on the categorical data have been devised, which are ROCK [17], CACTUS [18], COBWEB [19], to name a few. Unfortunately, this type of algorithms are basically rooted on the hierarchical clustering scheme where the time complexity cannot be lower than the quadratic time. Therefore, its computing time will be prohibitively long for a large spatio-textual dataset.

Mixed data clustering. Compared with spatial clustering algorithms or categorical clustering algorithms, there are only a few works on clustering mixed numeric and categorical data [20,21]. One of the most representative algorithms is the *k*-`prototypes` algorithm [20]. The *k*-`prototypes` can be used to perform spatio-textual clustering by representing the textual dimension of each object as a series of boolean values, one for one keyword. However, the *k*-`prototypes` algorithm and other clustering algorithms for mixed data share the strict assumption that every object has the same number of categorical attributes. This means, in spatio-textual data, the textual dimensionality of each object becomes extremely high, and therefore mixed data clustering algorithms including *k*-`prototypes` cannot efficiently work with a massive dataset.

3. Problem formulation

This section formally defines the problem environment and clarifies the goal of the paper.

Our problem environment follows the many works in the literature of spatio-textual similarity search [22,4,23], which is summarized as follows:

- We consider a set of spatio-textual objects, denoted by $O = \{o_1, o_2, \ldots, o_{|O|}\}$.
- Each object $o \in O$ consists of two attributes, namely $<loc, \tau>$, where loc is a geographic location and $\tau = \{t_1, t_2, \ldots, t_{|\tau|}\}$ is a set of keywords.
- Each keyword $t \in o.\tau$ is associated with a weight $w(t)$, which represents the significance of the keyword and is global for all objects. The most widely used value for the keyword weight is the inverted document frequency (idf).
- The distance between two spatio-textual objects o_1 and o_2 is defined as:

$$Dist(o_1, o_2) = \alpha \cdot DistS(o_1, o_2) + (1-\alpha) \cdot DistT(o_1, o_2),$$

where α is a user parameter to adjust the importance of spatial dimension or textual dimension, $DistS(*, *)$ is the Euclidean distance between $o_1.loc$ and $o_2.loc$, and $DistS(*, *)$ is the weighted Jaccard distance between $o_1.\tau$ and $o_2.\tau$ defined as follows:

$$1 - \frac{\sum_{t \in o_1.\tau \cap o_2.\tau} w(t)}{\sum_{t \in o_1.\tau \cup o_2.\tau} w(t)}.$$

Note that α is not a parameter to be optimized in advance, but rather it represents a user's intention on whether s/he is interested in the spatial aspect or the textual aspect of the underlying dataset.

Then our problem is formally defined as follows:

Definition 1. Given O, $DistS(*, *)$, and a positive integer k, partition O into k disjoint clusters such that the total intra cluster distance is minimized and the total inter cluster distance is maximized with respect to $DistS(*, *)$.

4. Spatio-textual K-means clustering

This section first introduces the basic flow of k-`means` as it is also the underlying clustering scheme of our algorithm, and then presents our proposed clustering algorithm that addresses the difficulty in applying k-`means` to spatio-textual data.

4.1. K-means

Let us first examine the k-`means` algorithm. k-`means` is basically the iterative process of finding k optimal centroids that minimize the sum of squared Euclidean distances between all objects and their closest centroid. At a high level, the process of k-`means` is summarized as Algorithm 1.

Algorithm 1. K-MEANS(O, k, $DistS(*, *)$).

```
Input $O$:= a set of objects, $k$:= the number of clusters, $DistS(*, *)$:= a distance measure between two objects
Output $\mathbb{C}$:= a set of $k$ disjoint clusters of $O$
1 Choose $k$ initial centroids;
2 repeat
3     $\mathbb{C} \leftarrow$ Assignment of all the objects to the closest centroid based on $DistS(*, *)$;
4     Update the centroid of each cluster based on $\mathbb{C}$;
5 no change in $\mathbb{C}$;
6 return $\mathbb{C}$
```

In finding spatio-textual clusters, the major difficulty of the above k-`means` algorithm lies in updating centroids. Thus, it is not intuitive to virtually create spatio-textual centroids since k-`means` basically assumes that every attribute is numeric while the textual dimension is defined as a set of keywords. Even though there can be several ways of converting such a set of keywords to a numeric value such as the term frequency vector, they commonly suffer from the sparse and high dimensional representation. In other words, the total number of keywords in the entire dataset must be huge, but each object contains only a small number of keywords compared with the entire set of keywords.

Unfortunately, a similar problem occurs when using the set representation for each textual centroid. When the cluster size grows, the number of keywords in the centroid unavoidably increases as the centroid should somehow contain the textual information for all the keywords in its corresponding cluster. This phenomenon is widely called "*ripple effect*", and known to make both the efficiency and quality of clustering worse [17].

Our solution remedies the above difficulty by removing virtual centroids in the process of k-`means` clustering. Instead, we utilize the *expected distance* between a random pair of objects. Our intuition is that the centroid is ultimately intended for finding the closest spatio-textual cluster for each object, and it suffices for this task to compute the expected distance of a random object in each cluster from the object under consideration.

Therefore, the overall flow of our query processing algorithm can be expressed as a more general version of Algorithm 1, which consists of the following three components:

1. Initialization (Line 1)
2. Assignment (Line 3)
3. Update (Line 4)

4.2. Expected distance based K-means

In this section, we propose our modified version of k-`means` algorithm for spatio-textual data by presenting our methods tackling each of the aforementioned steps.

4.2.1. Initialization

One of the most important issues in k-`means` family algorithms is how to choose initial k seeds. This is especially because that it is known that the quality and performance of k-`means` clusters are highly sensitive to the first k seeds [13]. The most straightforward approach is to choose arbitrarily k random seeds, but it turns out that it is the weakest point of the conventional k-`means` algorithm. Several attempts are proposed to more wisely choose k initial individual objects [13,6]. However, it is not a very good approach for spatio-textual data to use k individual objects as the initial status, especially due to the characteristics of the textual dimension. It is very common in textual dimension that only a few keywords are shared by different objects. Therefore, individual object cannot effectively represent other objects that can be combined into the same cluster.

For the initialization process, our approach is to choose k groups of objects, instead of k objects, as the initial k clusters. Of course, such k groups are regarded as small subsets of final k clusters. Thus, the objects in each group should be similar to each other textually as well as spatially.

To find such k spatio-textual *mini*-clusters, we utilize a grid-based space partitioning scheme and consider the textual cohesion of each cell. The underlying idea is that the objects residing in a cell can be seen *already spatially close* to the others in the same cell, and therefore it suffices to estimate their *textual cohesion*. However, a grid-based method has the following drawbacks as usual. First of all, the number of objects in each cell can be dramatically different, depending on how data points are distributed in space. In a statistical point of view, cell with only a few points cannot be regarded as equally significant as those with many points. The other problem is that it is not easy to accordingly adjust the length of each cell.

To overcome these drawbacks, we employ the partitioning scheme adapted from a quadtree [24], where the whole space is recursively split into four equal-sized quadrants. Basically, we divide the entire set of objects into four subsets in such a way that all the objects in each subset reside in one of four quadrants, and we do the same division process recursively for each subset. This process continues until each quadrant has at most a given number, called *the cell capacity*, of objects. We consider only those quadrants within the cell capacity to be the cells in a *multi-granularity* grid structure. By doing so, each cell has a similar number of objects even though their lengths will not be the same. Also, we no longer have to adjust the cell length.

If the textual cohesion of a cell is high enough compared to its cell length, we can think that the cell is one of the spatio-textual mini-clusters. The textual cohesion of each cell is defined as follows:

Definition 2. (**Textual Cohesion**) Let C be a cell. Then the textual cohesion of C is defined as the expected textual distance between a random pair of objects in C.Now we have to examine how to compute the expected textual distance between a random pair of objects in C. Instead of computing the exact expected textual distance, we use the estimation of the expected textual distance by employing the keyword frequency statistics of each cell. To this end, we maintain for each cell a hash table consisting of pairs of a keyword and its frequency in the cell, namely $<t_i, n_C(t_i)>$, where $n_C(t_i)$ is the number of occurrences of keyword t_i in C.

Then the expected textual distance between a random pair of objects o and o' is formally represented as:

$$E[DistT(o, o')] = 1 - E\left[\frac{\sum_{t \in o.\tau \cap o'.\tau} w(t)}{\sum_{t \in o.\tau \cup o'.\tau} w(t)}\right] \tag{1}$$

To estimate (1), let us first use L to denote the total weight of keywords of a random object $o \in C$, which is:

$$L = \sum_{t \in o.\tau} w(t) \tag{2}$$

Let W_C be the entire set of keywords in C, i.e., $\bigcup_{o \in C} o.\ \tau$. Then the expectation of L can be derived as:

$$E[L] = \sum_{t \in W_C} w(t) \cdot Pr(t \in o.\ \tau), \tag{3}$$

where $Pr(t \in o.\ \tau)$ is the probability for the occurrence of t in the textual dimension of o. Based on the hash table maintained by each cell, we use $n_C(t)/|C|$ for $Pr(t \in o.\ \tau)$. It is worth noting that the value of $E[L]$ can be easily computed by scanning C once.

We consider another random variable M that denotes the total weight of common keywords of a random pair of objects $o \in C$ and $o' \in C$, which is:

$$M = \sum_{t \in o.\tau \cap o'.\tau} w(t) \tag{4}$$

Similarly we have:

$$E[M] = \sum_{t \in W_C} w(t) \cdot Pr(t \in o.\ \tau \cap o'.\ \tau) \sum_{t \in W_C} w(t) \cdot Pr(t \in o.\ \tau) \cdot Pr(t \in o'.\ \tau \mid t \in o.\ \tau).$$

Similar to $Pr(t \in o.\ \tau)$, $Pr(t \in o'.\ \tau | t \in o.\ \tau)$ can be defined, by using the hash table of keyword frequencies, as $(n_C(t) - 1)/(|C| - 1)$, which is the probability of having t without replacement. Also, $E[M]$ can be obtained by the linear scan of C.

Then (1) can be estimated as:

$$1 - E\left[\frac{M}{2L - M}\right] \approx 1 - \frac{E[M]}{2E[L] - E[M]} = \widehat{E[DistT(o, o')]} \tag{5}$$

As specified in (5), we use $E[Dist\widehat{T}(o, o')]$ to denote the estimation of the textual distance between a random pair of objects o and o'. Considering the maximum spatial distance together with the textual cohesion, we choose top k cells with the highest $\alpha \cdot celldiameter + (1 - \alpha) \cdot E[Dist\widehat{T}(o, o')]$ values, and this sets of objects, one set for one cell, will be in charge of initial k mini-clusters.

The entire process of division step can be run in $O(|O| \log |O|)$ time since only linear scan of O is required for each quadrant split, and the number of splits is $O(\log |O|)$.

Analysis on estimation error. Theoretically, the derivation of $E[Dist\widehat{T}(o, o')]$ in (5) is not guaranteed to be 100% accurate in all cases in the sense that L and M may not be independent. However,

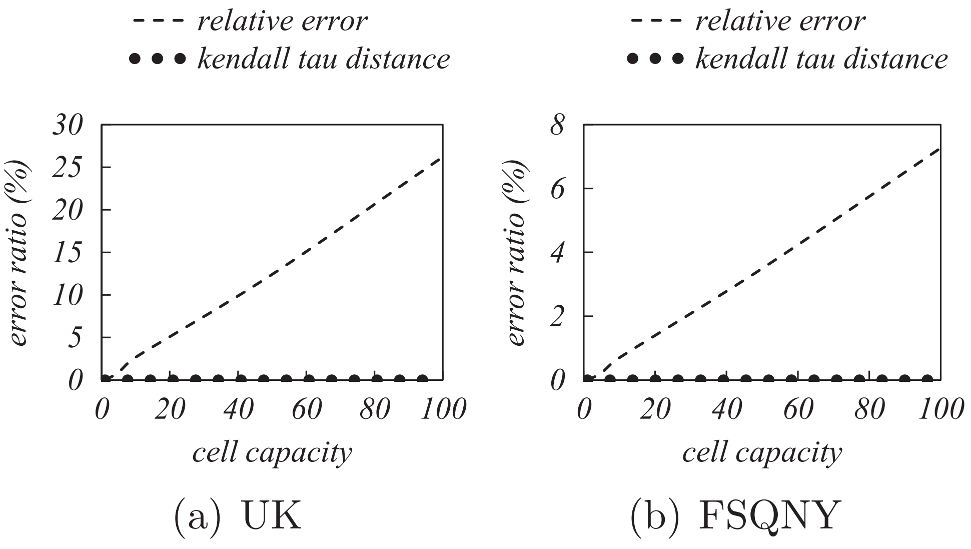

Fig. 1. Estimation error on textual cohesion. (a) UK (b) FSQNY.

heuristically we observed the fact that our estimation is accurate enough in practice especially when the cell capacity is not very large. Fig. 1 shows the average relative error ratio between $E[Dist\widehat{T}(o, o')]$ and $E[DistT(o, o')]$ over the set of all cells partitioning two real datasets namely `UK` and `FSQNY` (to be explained in Section 5.1), when varying the cell capacity. Overall, the relative error linearly increases as the cell capacity increases.

Furthermore, note that our aim of estimating the textual cohesion is not to obtain exact values themselves, but to find out which cell is textually tighter than the others. Thus, it suffices to obtain the same ordering on cells by using our estimation as the ordering obtained by the exact textual cohesion. In order to evaluate this, we use the normalized *kendall tau distance* that is a metric for comparing two ranking lists, where the distance values indicates the ratio of discordant pairs between two lists. We consider top-50 textually tightest cells for both of ordering lists, and measure the kendall tau distance between two lists in a way presented in [25]. As shown in Fig. 1, the kendall tau distance value stays always 0 even though its corresponding relative error gets higher as the cell capacity increases. It implies that, for the purpose of picking up top k initial seeds, our estimation does its job fairly well.

4.2.2. Assignment

Based on the initial k mini-clusters, we now have to assign all other objects to their closest cluster. Instead of finding the closest centroid in the normal k-`means` algorithm, our algorithm finds the cluster whose expected distance between a random object in the cluster and the object to be assigned is minimum.

Once again, we use the estimation of the expected distance rather than the exact value that can be obtained by averaging all distances of objects in the cluster from the object being assigned.

Since our distance metric consists of a spatial part and a textual part, we need to estimate each of them individually and combine those two estimations. Let us first consider estimating the expected textual distance between a cluster and the object being assigned. The process is quite similar to estimating the textual cohesion of a cluster. Let q be the object to be assigned, and let C be a cluster (it was a cell). Then for a random object o in C, $E[DistT(o, q)]$ can be represented as:

$$1 - E\left[\frac{\sum_{t\in o.\tau\cap q.\tau} w(t)}{\sum_{t\in o.\tau} w(t) + \sum_{t\in q.\tau} w(t) - \sum_{t\in o.\tau\cap q.\tau} w(t)}\right] \tag{6}$$

The difference of (6) from (1) is that q is not a random object, and hence $\sum_{t\in q.\tau} w(t)$ is just a constant. Since $\sum_{t\in o.\tau} w(t)$ is the same as (2), denoted by L, all we have to do is estimate the expectation of $\sum_{t\in o.\tau\cap q.\tau} w(t)$. Let Z denote $\sum_{t\in o.\tau\cap q.\tau} w(t)$. Then $E[Z]$ can be represented as:

$$E[Z] = \sum_{t\in q.\tau} w(t)\cdot Pr(t \in o.\ \tau). \tag{7}$$

The hash table of C enables, for any term t, that $Pr(t \in o.\ \tau)$ can be obtained in a constant time, and therefore $E[Z]$ can be computed in $O(|q.\ \tau|)$ time.

Note that since we do not use any textual centroid, we can reduce the cost of calculating distances between the object being assigned and the textual centroid that can be very long for a large-sized cluster.

By (7), we finally estimate (6) as follows:

$$\begin{aligned}1 - E\left[\frac{Z}{L + \sum_{t\in q.\tau} w(t) - Z}\right] &\approx 1 - \frac{E[Z]}{E[L] + \sum_{t\in q.\tau} w(t) - E[Z]}\\ &= E[Dist\widehat{T}(o, q)]\end{aligned} \tag{8}$$

In order to estimate the expected spatial distance, namely $E[DistS(o, q)]$, we simply use the spatial centroid as follows:

$$\begin{aligned}E[DistS(o, q)] &= E[\sqrt{(o.\ loc.\ x - q.\ loc.\ x)^2 + (o.\ loc.\ y - q.\ loc.\ y)^2}]\\ &\approx \sqrt{(E[o.\ loc.\ x] - q.\ loc.\ x)^2 + (E[o.\ loc.\ y] - q.\ loc.\ y)^2}\\ &= E[Dist\widehat{S}(o, q)]\end{aligned} \tag{9}$$

Note that this is exactly the same as we do in the normal k-`means` algorithm.

By combining (8) and (9) using α, we can finally define the distance between a cluster C and the object q to be assigned as follows:

$$Dist(C, q) = \alpha\cdot E[Dis\hat{t}S(o, q)] + (1 - \alpha)\cdot E[Dist\widehat{T}(o, q)]$$

For each object q, we assign q to the cluster C with the smallest $Dist(C, q)$ among k clusters.

4.2.3. Update

In the normal k-`means` algorithm, after assigning all the objects, we re-compute all k centroids for updated k clusters. In our approach, we adopt an incremental updating scheme to improve the efficiency. For each cluster C, the information to be updated is summarized as follows:

- The hash table containing all the keywords in C
- The textual cohesion of C
- The spatial centroid of C, namely $E[o.\ loc]$

The hash table is required for obtaining $E[Z]$ as shown in (7). Also, for both (5) and (8), we should have $E[L]$ to be updated. Similarly, $E[M]$ is also essential to compute the textual cohesion. Finally, for (9), we have to keep $E[o.\ loc]$ as well.

Note that since every assignment can entail updating the above values, the time complexity of each update should be a constant. It is obvious to incrementally update the hash table, $E[L]$, and $E[o.\ loc]$ in a constant time. However, updating $E[M]$ deserves a bit more explanation.

In our formula, $E[M]$ is represented as:

$$\begin{aligned}E[M] &= \sum_{t\in W_C} w(t)\cdot Pr(t \in o.\ \tau)\cdot Pr(t \in o'.\ \tau | t \in o.\ \tau)\\ &= \sum_{t\in W_C} w(t)\cdot\frac{n_C(t)}{|C|}\cdot\frac{n_C(t) - 1}{|C| - 1}\end{aligned} \tag{10}$$

Based on (10), we can observe that it suffices to maintain the value of $\sum_{t\in W_C} w(t)\cdot n_C(t)\cdot(n_C(t) - 1)$, and thereby (10) can be obtained in a constant time divided by $|C|\cdot(|C| - 1)$ whenever required. Therefore, for each term t_q of the object q being assigned, we subtract the previous value for t_q, i.e., $w(t_q)\cdot n_C(t_q)\cdot(n_C(t_q) - 1)$, and add the updated value, $w(t_q)\cdot n_C(t_q)\cdot(n_C(t_q) + 1)$. This leads to simply adding $2\cdot w(t_q)\cdot n_C(t_q)$ for each arrival of t_q. Similarly, whenever an object q is escaped from the cluster C, we can subtract $2\cdot w(t_q)\cdot n_C(t_q)$ for each term t_q in $q.\ \tau$.

It is worth noting that the side benefit of our incremental updating scheme is that even during the assignment phase, the quality of clustering is gradually improved. Thus, the up-to-date cluster information is likely to help the next object to be assigned to make the better choice.

4.2.4. Overall algorithm

Combining all detailed processes, we present the overall algorithm in Algorithm 2. We first divide O spatially into multiple quadrants (i.e., cells) and choose k mini-clusters, each of which has the smallest weighted sum of the textual cohesion and the diameter, in the initialization phase (Lines 2–10). After that, for each object, we assign it to the closest cluster in terms of the expected spatio-textual distance (Lines 12–13). Whenever the assignment task is finished, we accordingly update all the statistics and information for each cluster (Lines 14–17).

Algorithm 2. EXP-K-MEANS(O, k, $DistS(*, *)$).

```
Input: $O$ := a set of objects, $k$ := the number of clusters, $Dist(*,*)$ := a
       distance measure between two objects
Output: $\mathbb{C}$ := a set of $k$ disjoint clusters of $O$
1  $\mathbb{C} \leftarrow \emptyset$;
2  Divide $O$ spatially into the set of cells in a multi-granularity grid $G$;
3  foreach cell $C \in G$ do
4      if $|\mathbb{C}| < k$ then
5          Insert $C$ into $\mathbb{C}$ as a seed;
6      else
7          $tc(C) \leftarrow$ Compute the textual cohesion of $C$;
8          $dia(C) \leftarrow$ The length of diagonal of $C$;
9          if $\alpha \cdot dia(C) + (1-\alpha) \cdot tc(C)$ is the smallest so far in all seeds in $\mathbb{C}$
             then
10             Insert $C$ into $\mathbb{C}$ as a seed and remove the one with the smallest
               textual cohesion from $\mathbb{C}$;
11 repeat
12     foreach $o \in O$ do
13         Assign $o$ to the cluster in $\mathbb{C}$ whose expected spatio-textual distance
             from $o$ is minimum.
14     foreach $C \in \mathbb{C}$ do
15         Update the hash table containing all keywords in $C$;
16         Update the textual cohesion of $C$;
17         Update the spatial centroid of $C$;
18 until no change in $\mathbb{C}$;
19 return $\mathbb{C}$
```

We can prove that Algorithm 2 is always guaranteed to converge as follows:

Theorem 1. Algorithm 2 terminates after a finite number of iterations.

Proof. Consider an iterative step. When we re-assign each object to a new cluster, we choose one with the minimum distance (i.e., $\min_{i=1\ldots k} Dist(o, C_i)$ as defined in Equation (10)). Therefore, the total distance of all objects from their closest cluster should decrease after any iteration and can never increase. Since the algorithm stops when we cannot find any better clustering whose total distance is smaller than that of the previous one, it is guaranteed to terminate at a particular state.

Complexity analysis. Now let us analyze the time complexity of our clustering algorithm. We use N to denote $|O|$ and ℓ to denote the average number of keywords constituting the textual dimension of an object in O, i.e., $\ell = \frac{\sum_{o \in O} |o.\tau|}{N}$. Also, let m denote the number of iterations.

- **Initialization part** (Lines 2–10)
  In the initial seeding phase, we first divide the entire dataset into the set of quadrants, which entails $O(N \log N)$ time. Also, we have to scan the subset corresponding to each cell to construct a keyword map for the cell, which can be run in total $O(\ell N)$ time. Choosing the initial k cells can be done in $O(k|G|)$, where $|G|$ is the number of cells in our space partitioning scheme. Usually, $|G|$ is much smaller than N, and therefore the cost of the initial phase can be determined by the division step and the construction of the grid-based keyword map, that is, $O(N \log N + \ell N)$.
- **One iteration part** (Lines 11–16)
  In the assignment phase (Lines 11–12), each object needs to be examined to determine which cluster is the closest among k clusters. This cost is $O(k\ell)$ (recall that the formula (7) can be calculated in $O(|o.\tau|)$ for any object $o \in O$), and hence the entire cost of the assignment step is $O(k\ell N)$. Thanks to our incremental updating scheme, the updating task can be done in a piggybacking manner. This implies that the overall cost of one iteration is dominated by the cost of the assignment step, that is, $O(k\ell N)$.
- **Total complexity**
  The total cost of all iterations is $O(k\ell Nm)$, and this dominates the cost of initial seeding since $O(\log N)$ is not higher than $O(k\ell m)$ in most cases.

Note that this complexity is far lower than those of alternative k-partitioning algorithms such as k-`medoids` and k-`prototypes`. The time complexity of one iteration of k-`medoids` is known as $O(k(N-k)^2)$ [16], and it becomes $O(k\ell(N-k)^2)$ in our spatio-textual environment because the cost of calculating the textual distance is $O(\ell)$. Therefore, considering the iteration counts, its total complexity ends up $O(k\ell(N-k)^2 m)$, which is a factor of N times higher than that of our algorithm.

In the case of k-`prototypes`, the one iteration cost can be represented as $O(kLN)$, where L is the average number of unique keywords in each cluster, since for each object we compute the distances from k centroids (i.e., prototypes) [20]. Note that L can be as large as $O(\ell N)$ when the cluster size gets large. In those worst cases, the running time of k-`prototypes` is most likely to be close to $O(k\ell N^2 m)$, which is also a factor of N times higher than ours.

In summary, our spatio-textual clustering algorithm can theoretically obtain a $O(N)$ speedup compared with alternative k-partitioning algorithms.

5. Experiments

In this section, we focus on the following two questions. One is to experimentally evaluate how much faster our clustering algorithm is compared with alternative algorithms. The other is to check whether such a speed-up negatively affects the quality of resulting clusters. Lastly, with respect to these two questions, we examine the benefits of our seeding technique exploiting the grid structure compared with the simple random seeding method.

5.1. Environments

Parameters. There are two important parameters in our problem, which are k and α. As used throughout the paper, k is the number of clusters and α is the user parameter to adjust the importance of the spatial aspect and the textual aspect. The details including default values are presented in Table 1.

Datasets. To evaluate the performance of our algorithm in practice, we use two real datasets. The first one is `UK`[3], which is the set of POIs (e.g., hospital, supermarket, park, etc.) of the United Kingdom, where each POI is augmented with a simple textual description. It contains about one million words, and its number of unique words is about 0.1 million.

The other real dataset is `FSQNY` collected from 0.5 million venues of *Foursquare*[4] in New York, USA. Its total number of words and total number of unique words are about 2.4 million and 0.16 million, respectively. Further details are shown in Table 2.

All the spatial coordinates are normalized for the pairwise spatial distance to be at most 1 as the pairwise textual distance is in the range of [0, 1].

For the weight of each keyword t, we adopt the concept of RSJ (Robertson-Sparck Jones) weight, which was shown to be more effective than the commonly-used Inverse Document Frequency (IDF) weights [26]. More specifically, we use the following rough version of the RSJ weight definition:

$$w(t) = \log\frac{W - f(t) + 0.5}{f(t) + 0.5}$$

where W is the total number of words and $f(t)$ is the number of occurrence of the keyword t.

Competitors. We implement the following clustering algorithms and compare their performances in terms of the efficiency and the clustering quality:

- `PAM` [15] - As a baseline algorithm, we use the very first version of the k-`medoids` algorithm. In fact, when we say k-`medoids`, it usually refers to this algorithm. Despite its simplicity, it is generally known as not practical due to the high complexity.
- `CLARANS` [16] - This is a famous clustering algorithm intended to overcome the high complexity of `PAM`. The algorithm utilizes a sampling concept to reduce the running time. Instead of checking all objects in the neighborhood, it examine only a random sample set of neighbors. By doing this, `CLARANS` is generally accepted as the fastest derivative algorithm from k-`medoids`. We use the same experimental parameter value as used in [16] such as the percentage of neighbors examined being 1.25%.
- `KPrototype` [20] - As mentioned earlier, this is an extended k-`means` algorithm to deal with a mixed dataset where each object consists of both numeric and categorical attributes. By considering a set of keywords as a set of categorical values, this algorithm can handle the spatio-textual data as well. Since it is not reasonable and sometimes infeasible to regard all infrequent keywords as categorical values, we use only frequent keywords whose frequency is at least 1% of $|O|$. The number of those frequent keywords of `UK` and `FSQNY` are presented in Table 2.
- `ExpKMeans`- This is our proposed algorithm. It takes one parameter, namely the cell capacity. As shown in Fig. 1, any value from 0 to 100 would work well for the estimation. We fix the cell capacity to 10 in our experiments.

[3] http://www.pocketgpsworld.com
[4] https://foursquare.com

Table 1
The values of parameters.

Parameter	Values **(default value)**
Number of clusters (k)	2, 4, **(6)**, 8, 10
Adjusting parameter (α)	0.3, 0.5, **(0.7)**, 0.9

Table 2
The properties of real datasets.

Dataset	`UK`	`FSQNY`
Number of objects	179,491	512,590
Number of words	1,167,018	2,404,029
Number of unique words	117,305	166,909
Number of words with frequency higher than 1%	83	64

The main reason behind the selection of these competitors is that they are considered as the most efficient and scalable ones in their branch of works, except for `PAM`. This is also consistent to our main purpose, that is, developing a scalable and efficient clustering algorithm for massive spatio-textual datasets prevalent in many location-based systems these days. Note that we do not use any dimensionality reduction techniques in the literature of document clustering methods. Those techniques are transparently applicable to all the competitors as well as our clustering algorithm.

5.2. Efficiency

In order to evaluate the efficiency of our algorithm, we compare the execution times of all the algorithms by varying parameter values.

Unfortunately, `PAM` and `KPrototype` were way too slow to process both of our real datasets, and therefore we had to use smaller datasets to see the overall performance comparison. To this end, we randomly sample both `UK` and `FSQNY` with a probability 1%, and thereby obtain smaller version of `UK` and `FSQNY` datasets roughly having 0.01 times their original cardinalities. We perform all the algorithms on 10 different samples and take the average of measurements. Figs. 2 and 3 show the results using those samples over two real datasets.

The execution time tends to get longer when k increases. It is obvious that a larger k increases the execution time as analyzed in Section 4.2.4. Also, a small α tends to particularly increase the execution time of `KPrototype`. This implies that the textual dimension makes `KPrototype` more difficult to converge since our implementation of `KPrototype` does not deal with infrequent keywords which many cause some loss of information.

Considering that the execution time is presented in the log scale, we can see that our algorithm is superior to the other algorithms in terms of the efficiency. Our algorithm is almost two orders of magnitude faster than even `CLARANS`, which is the fastest algorithm among competitors.

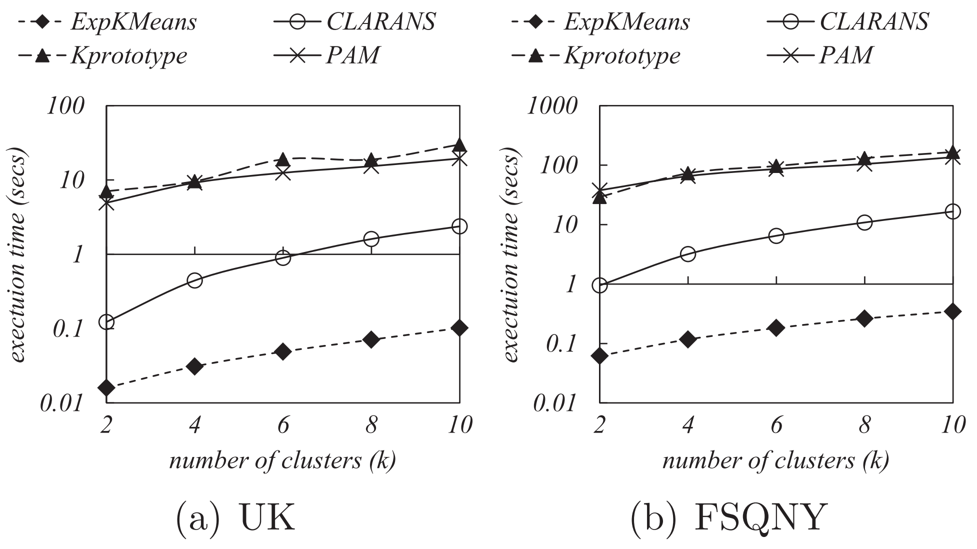

Fig. 2. Efficiency test with varying k (sampled). (a) UK (b) FSQNY.

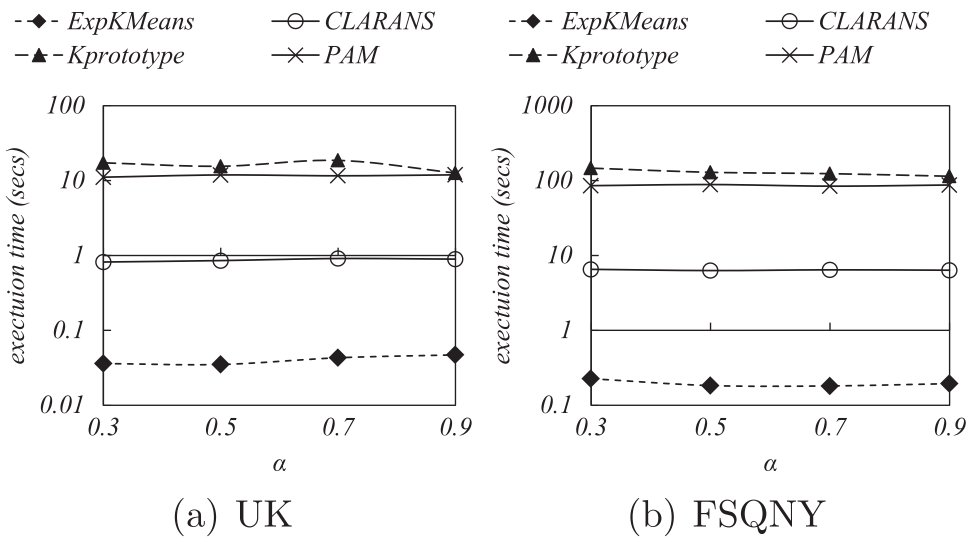

Fig. 3. Efficiency test with varying α (sampled). (a) UK (b) FSQNY.

5.3. Scalability

In order to test our algorithm in a more scalable environment, we also conduct experiments using larger samples of real datasets by comparing only with CLARANS. As shown in Fig. 4, ExpKMeans achieves much higher scalability than CLARANS. When the sample size gets larger, the execution time of CLARANS becomes extremely longer to such an extent that it was not even able to measure the execution time for the original FSQNY dataset. We can observe that only our algorithm is scalable enough to process two original real datasets in a reasonable time.

5.4. Clustering quality

In order to measure the quality of clustering results, we use the *silhouette coefficient*, which is widely used especially when the ground truth of a dataset is not available [27]. For each object $o \in O$, its silhouette coefficient is defined as:

$$s(o) = \frac{b(o) - a(o)}{max\{a(o), b(o)\}}$$

where $a(o)$ is the average distance between o and all other objects in the cluster to which o belongs, and $b(o)$ is the minimum average distance from o to all other objects in clusters to which o does not belong. The $s(o)$ value is in the range of $[-1, 1]$. When the value of $s(o)$ gets close to 1, it means that o is currently assigned to an appropriate cluster.

In order to evaluate the quality of a clustering, we can calculate the average over all $s(o)$ values of objects in O. Unfortunately, computing $a(o)$ and $b(o)$ requires a substantial amount of time. Therefore, in our experiments, we adopt the idea of CLARANS to avoid such a long evaluating time. Instead of examining all the other objects, we consider a smaller sample set of objects when computing both $a(o)$ and $b(o)$. For a sampling parameter, we use 0.1 for sampled datasets, and 0.001 for original datasets.

Figs. 5 and 6 show the (estimated) average silhouette coefficients of all objects using 10 different samples of real datasets. The higher the value, the better the clustering. Overall, the qualities of k-means derivative algorithms (i.e., KPrototype and ExpKMeans) tend to be better than those of k-medoids and its variant (i.e., PAM and CLARANS). This is probably due to the fact that a single median object cannot effectively represent its cluster because of the sparsity of textual data.

In all the results, our algorithm is almost always at the highest position together with KPrototype in terms of the clustering quality. Interestingly, it is observed that KPrototype tends to slightly get better with a lager α value but slightly get worse with a smaller α value, particularly compared to ExpKMeans. This implies that KPrototype

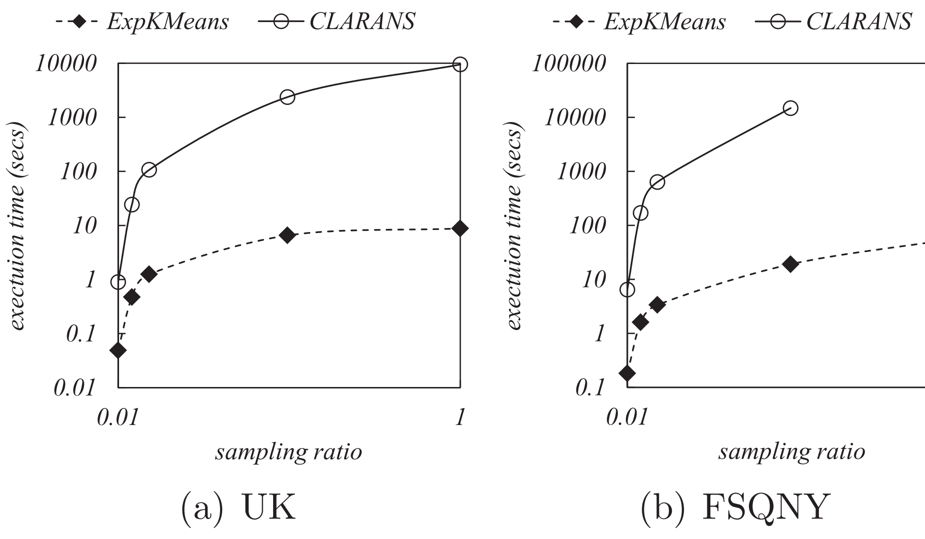

Fig. 4. Scalability test with varying the sampling ratio. (a) UK (b) FSQNY.

more focuses on the spatial dimension rather than the textual dimension. When α gets close to 1.0, `KPrototype` will work in almost the same was as k-`means` in that the resulting clustering will not be affected by textual distances even though we cannot still avoid computing all those textual distances. Based on this, we can conclude that our clustering scheme is not only efficient but also quite effective to catch the textual dimension for the clustering.

5.5. Benefit of the grid-based seeding method

In this section, we examine how our grid-based seeding method (explained in Section 4.2.1) is effective in practice. To this end, we use two versions of our algorithm; one is with the grid-based seeding technique, namely *Grid seeding*, and the other is with the simple random seeding scheme, namely *Random seeding*. In order to see how bad *Random seeding* could be in the worst case and in the average case, we conduct the same experiment 10 times and take its best, worst, and average.

Let us first see how the seeding method affects the execution time. In this series of experiments, we also consider some larger k values as well as k values used so far. This is because when k becomes large, the random selection technique is not expected to effectively catch the underlying k clusters in the first place. We also set the cell capacity here to 100 for processing our larger real datasets with larger k values.

Fig. 7 shows the elapsed times of *Grid seeding* and *Random seeding*, where the best case and the worst case in *Random seeding* are depicted as the error bar. In the average case of *Random seeding*, the execution time differences between *Grid seeding* and *Random seeding* are not dramatically large. In a few cases, the *Random seeding* shows a better performance than *Grid seeding*. However, when k increases, the gap between its best time value and its worst time value becomes extremely larger, and *Random seeding* ends up with a very bad result in the worst case, compared to *Grid seeding*.

The main underlying reason is that the number of iterations is highly sensitive to which objects are selected as initial seeds in performing *Random seeding* particularly when we consider a larger k value. On the other hand, *Grid seeding* shows a relatively robust and stable performance for larger k values because its initial k mini-clusters can effectively catch the underlying k clusters to be revealed.

Fig. 8 shows the result of the quality test. At this time, the gap between the worst and the best in *Random seeding* is not very large even with larger k values. It seems that both *Grid seeding* and *Random seeding* eventually succeed in finding the (locally) optimal clustering for a given k value. Therefore, their results are almost the same in terms of the clustering quality even though *Grid seeding* is a little bit better than *Random seeding* in most cases.

Fig. 9 shows the average value of all the experiments in this section, which effectively present that our proposed seeding technique is effective to reduce the overall execution time with preserving the quality.

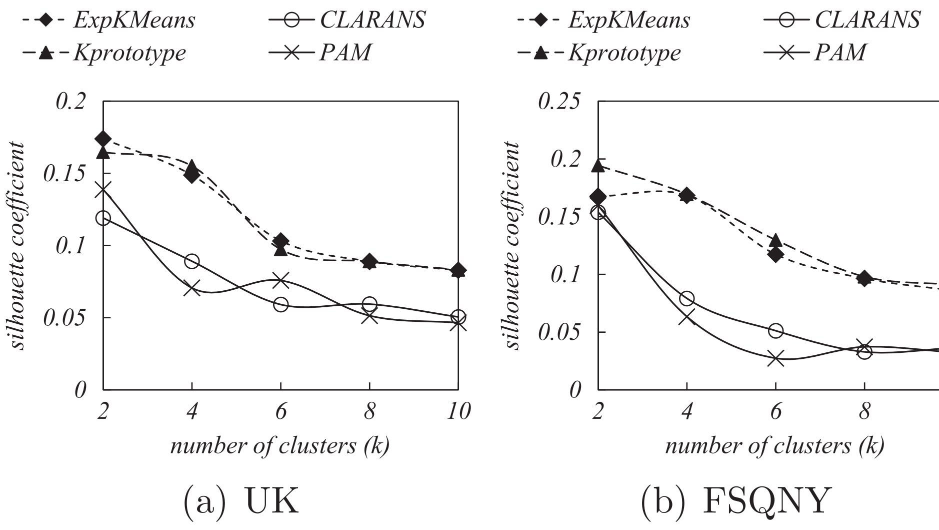

Fig. 5. Quality test with varying k (sampled). (a) UK (b) FSQNY.

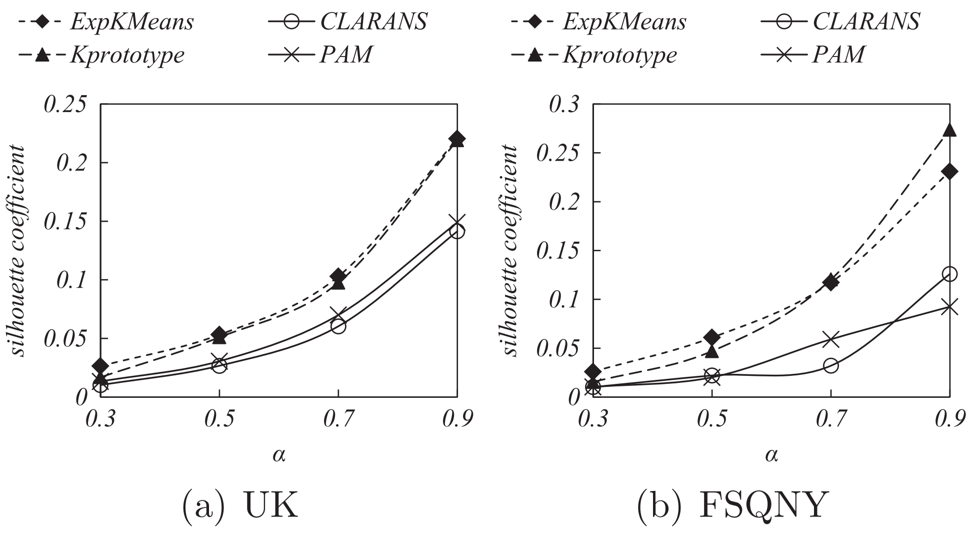

Fig. 6. Quality test with varying α (sampled). (a) UK (b) FSQNY.

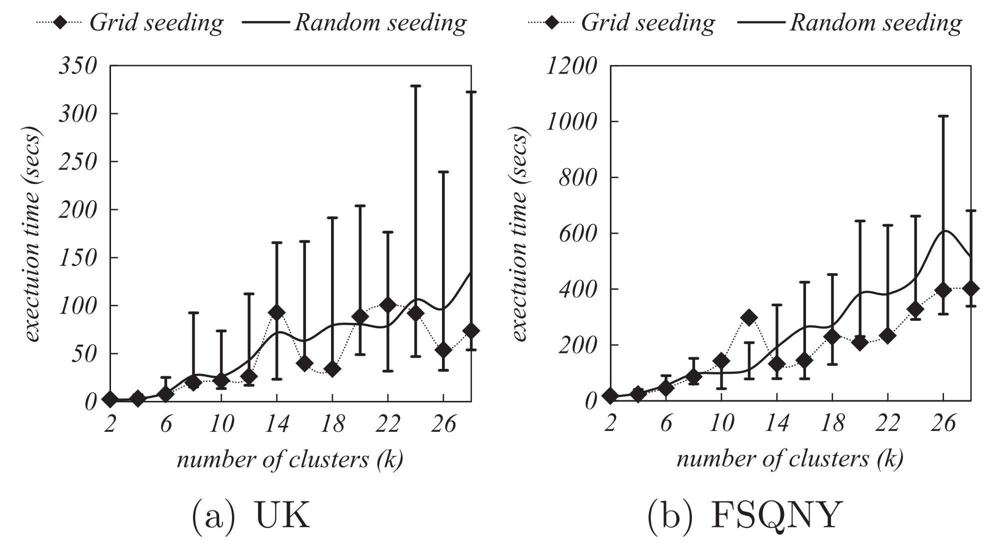

Fig. 7. Efficiency test of seeding methods. (a) UK (b) FSQNY.

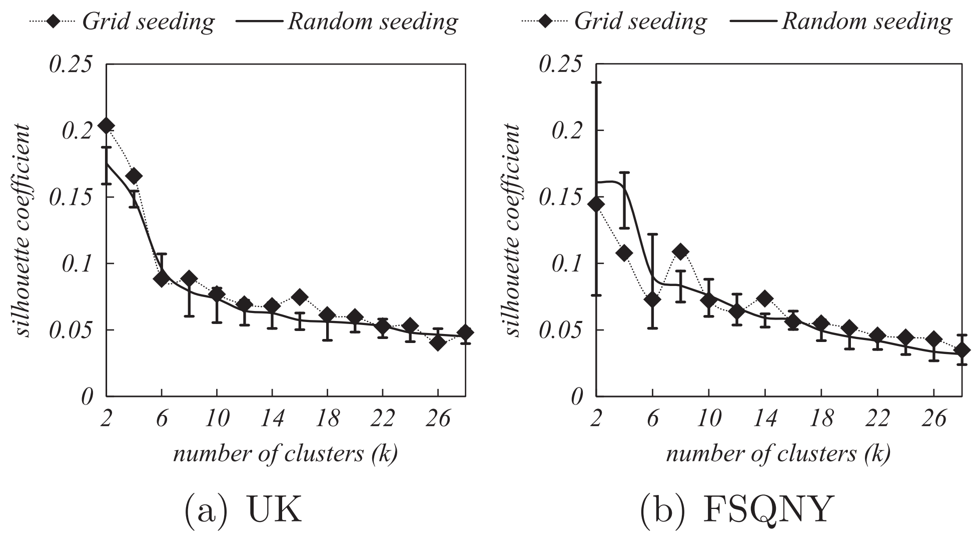

Fig. 8. Quality test of seeding methods. (a) UK (b) FSQNY.

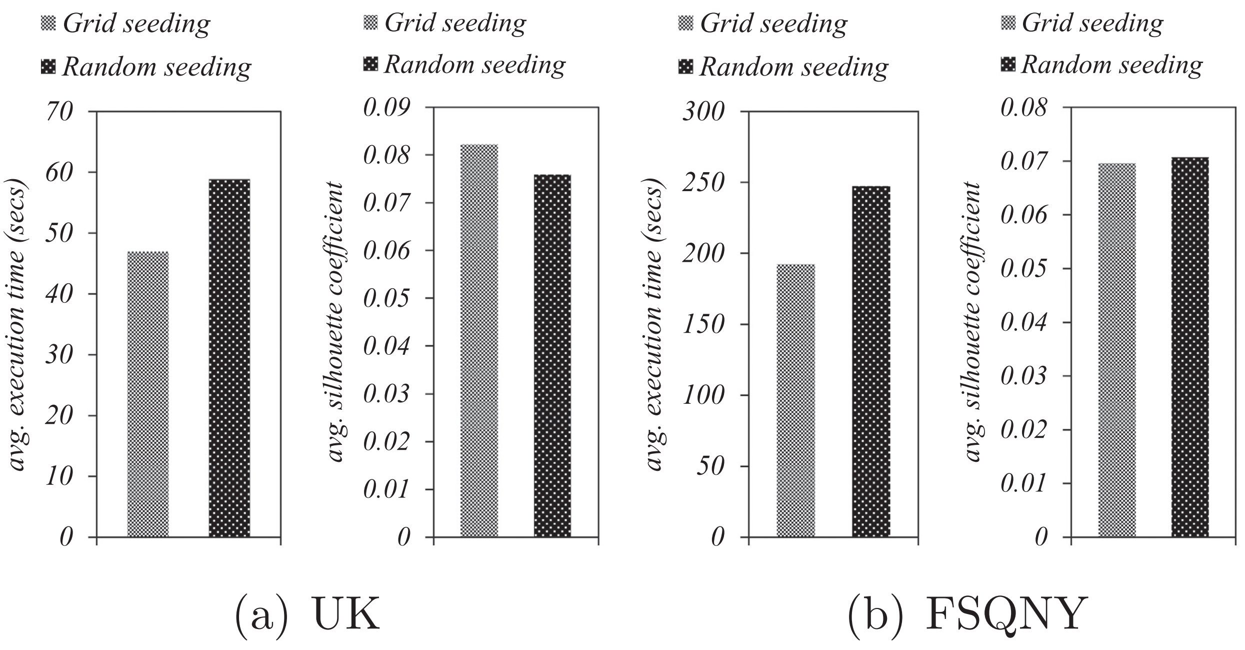

Fig. 9. Overall performance summary of seeding methods. (a) UK (b) FSQNY.

6. Conclusions

In this paper, we dealt with the spatio-textual data as a clustering problem, and developed a *k*-means family algorithm for clustering a large-scale spatio-textual dataset. In order to address the challenges in applying *k*-means to the spatio-textual data domain, we utilized the concept of the expected distance between two random objects (or one random object and one fixed object). By doing so, we have shown that spatio-textual clustering can be done in a very fast manner while preserving the reasonable clustering quality.

As a concluding remark, this paper has only considered the Jaccard distance for textual dimension, and it still remains a future work to estimate the expected other kinds of textual distances such as the cosine similarity. Our next plan is to generalize our clustering scheme to work with any kinds of textual distances.

Acknowledgements

We would like to thank anonymous reviewers and the editor. This work was supported in part by 2016 Seed Money Project of Chongqing Liangjiang KAIST International Program, Chongqing University of Technology, and in part by National Natural Science Foundation of China (Grant No. 61502064).

References

[1] I.D. Felipe, V. Hristidis, N. Rishe, Keyword search on spatial databases, in: Proceedings of the 24th International Conference on Data Engineering, ICDE, 2008, pp. 656–665.
[2] G. Cong, C.S. Jensen, D. Wu, Efficient retrieval of the top-k most relevant spatial web objects, PVLDB 2 (1) (2009) 337–348.
[3] B. Yao, F. Li, M. Hadjieleftheriou, K. Hou, Approximate string search in spatial databases, in: Proceedings of the 26th International Conference on Data Engineering, ICDE, 2010, pp. 545–556.
[4] J. Fan, G. Li, L. Zhou, S. Chen, J. Hu, SEAL: spatio-textual similarity search, PVLDB 5 (9) (2012) 824–835.
[5] Y. Tao, C. Sheng, Fast nearest neighbor search with keywords, IEEE Trans. Knowl. Data Eng. 26 (4) (2014) 878–888.
[6] B. Bahmani, B. Moseley, A. Vattani, R. Kumar, S. Vassilvitskii, Scalable k-means, PVLDB 5 (7) (2012) 622–633.
[7] X. Cao, L. Chen, G. Cong, C.S. Jensen, Q. Qu, A. Skovsgaard, D. Wu, M.L. Yiu, Spatial keyword querying, in: Conceptual Modeling - 31st International Conference ER 2012, Florence, Italy, October 15-18, 2012. Proceedings, 2012, pp. 16–29.
[8] J.A. Hartigan, M.A. Wong, Algorithm as 136: a k-means clustering algorithm, J. R. Stat. Soc. Ser. C (Appl. Stat.) 28 (1) (1979) 100–108.
[9] S.P. Lloyd, Least squares quantization in PCM, IEEE Trans. Inf. Theory 28 (2) (1982) 129–136.
[10] T. Zhang, R. Ramakrishnan, M. Livny, BIRCH: an efficient data clustering method for very large databases, in: Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data, 1996, pp. 103–114.
[11] M. Ester, H. Kriegel, J. Sander, X. Xu, A density-based algorithm for discovering clusters in large spatial databases with noise, in: Proceedings of the Second International Conference on Knowledge Discovery and Data Mining KDD, 1996, pp. 226–231.
[12] M. Ankerst, M.M. Breunig, H. Kriegel, J. Sander, OPTICS: ordering points to identify the clustering structure, in: SIGMOD 1999, Proceedings ACM SIGMOD International Conference on Management of Data, 1999, pp. 49–60.
[13] D. Arthur, S. Vassilvitskii, k-means++: the advantages of careful seeding, in: Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2007, New Orleans, Louisiana, USA, January 7–9, 2007, 2007, pp. 1027–1035.
[14] J. Shi, N. Mamoulis, D. Wu, D.W. Cheung, Density-based place clustering in geo-social networks, in: International Conference on Management of Data, SIGMOD, 2014, pp. 99–110.
[15] L. Kaufman, P. Rousseeuw, Finding Groups in Data: An Introduction to Cluster Analysis, John Wiley & Sons, 1990.
[16] R.T. Ng, J. Han, CLARANS: a method for clustering objects for spatial data mining, IEEE Trans. Knowl. Data Eng. 14 (5) (2002) 1003–1016.
[17] S. Guha, R. Rastogi, K. Shim, ROCK: A robust clustering algorithm for categorical attributes, in: Proceedings of the 15th International Conference on Data Engineering, ICDE, 1999, pp. 512–521.
[18] V. Ganti, J. Gehrke, R. Ramakrishnan, CACTUS- clustering categorical data using summaries, in: Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 1999, pp. 73–83.
[19] D.H. Fisher, Knowledge acquisition via incremental conceptual clustering, Mach. Learn. 2 (2) (1987) 139–172.
[20] Z. Huang, Extensions to the k-means algorithm for clustering large data sets with categorical values, Data Min. Knowl. Discov. 2 (3) (1998) 283–304.
[21] A. Ahmad, L. Dey, A k-mean clustering algorithm for mixed numeric and categorical data, Data Knowl. Eng. 63 (2) (2007) 503–527.
[22] P. Bouros, S. Ge, N. Mamoulis, Spatio-textual similarity joins, PVLDB 6 (1) (2012) 1–12.
[23] Y. Lu, J. Lu, G. Cong, W. Wu, C. Shahabi, Efficient algorithms and cost models for reverse spatial-keyword k-nearest neighbor search, ACM Trans. Database Syst. 39 (2) (2014) 13.
[24] H. Samet, R.E. Webber, Storing a collection of polygons using quadtrees, ACM Trans. Graph. 4 (3) (1985) 182–222.
[25] R. Fagin, R. Kumar, D. Sivakumar, Comparing top k lists, SIAM J. Discret. Math. 17 (1) (2003) 134–160.
[26] O. Hassanzadeh, M. Sadoghi, R.J. Miller, Accuracy of approximate string joins using grams, in: Proceedings of the Fifth International Workshop on Quality in Databases, QDB 2007, at the VLDB 2007 conference, 2007, pp. 11–18.
[27] J. Han, M. Kamber, J. Pei, 10 - cluster analysis: Basic concepts and methods, in: J. H. Kamber, J. Pei (Eds.), Data Mining (Third Edition), third edition Edition, Morgan Kaufmann, 2012, pp. 443–495.