A Constant Approximation Algorithm for Sequential Random-Order No-Substitution k-Median Clustering

Tom Hess
Department of Computer Science
Ben-Gurion University of the Negev
Beer-Sheva, Israel
tomhe@post.bgu.ac.il

Michal Moshkovitz
Department of Computer Science
Tel-Aviv University*
Tel Aviv, Israel
mmoshkovitz@eng.ucsd.edu

Sivan Sabato
Department of Computer Science
Ben-Gurion University of the Negev
Beer-Sheva, Israel
sabatos@cs.bgu.ac.il

Abstract

We study k-median clustering under the sequential no-substitution setting. In this setting, a data stream is sequentially observed, and some of the points are selected by the algorithm as cluster centers. However, a point can be selected as a center only immediately after it is observed, before observing the next point. In addition, a selected center cannot be substituted later. We give the first algorithm for this setting that obtains a constant approximation factor on the optimal cost under a random arrival order, an exponential improvement over previous work. This is also the first constant approximation guarantee that holds without any structural assumptions on the input data. Moreover, the number of selected centers is only quasi-linear in k. Our algorithm and analysis are based on a careful cost estimation that avoids outliers, a new concept of a linear bin division, and a multiscale approach to center selection.

1 Introduction

Clustering is a fundamental unsupervised learning task used for various applications, such as anomaly detection [Leung and Leckie, 2005], recommender systems [Shepitsen et al., 2008] and cancer diagnosis [Zheng et al., 2014]. In recent years, the problem of *sequential clustering* has been actively studied, motivated by applications in which data arrives sequentially, such as online recommender systems [Nasraoui et al., 2007] and online community detection [Aggarwal, 2003].

In this work, we study k-median clustering in the sequential *no-substitution* setting, a term first introduced in Hess and Sabato [2020]. In this setting, a stream of data points is sequentially observed, and some of these points are selected by the algorithm as cluster centers. However, a point can be selected as a center only immediately after it is observed, before observing the next point. In addition, a selected center cannot be substituted later. This setting is motivated by applications in which center selection is mapped to a real-world irreversible action. For instance, consider a stream of website users, where the goal is to instantaneously identify users that serve as social cluster centers and provide them with a promotional gift while they are still on the website. As another example, consider recruiting representative participants for a clinical trial out of a stream of incoming patients.

*This work was done while the author was at the University of California San Diego.

35th Conference on Neural Information Processing Systems (NeurIPS 2021).

The goal in the no-substitution k-median setting is to obtain a near-optimal k-median cost value, while selecting a number of centers that is as close as possible to k. For an adversarially ordered stream, it has been shown [Moshkovitz, 2021] that an algorithm that selects a number of centers that is sublinear in the stream length cannot obtain a constant approximation guarantee, unless some structural assumptions are imposed on the input data.

In this work, we show that in contrast, when the stream order is random, a constant approximation guarantee can be obtained without imposing structural assumptions on the data. Previous works on the random-order setting provide either an algorithm with an almost-constant approximation assuming a bounded metric space [Hess and Sabato, 2020] or an algorithm with an approximation factor that is exponential in k [Moshkovitz, 2021]. Thus, our guarantees are stronger and more general than the previous guarantees. Moreover, the number of centers selected by our algorithm is only quasi-linear in k and does not depend on the stream length.

Main result. We propose a new algorithm, `MUNSC` (Multiscale No-Substitution Clustering), that obtains a constant approximation factor with probability $1-\delta$ over the random order of the stream, while selecting only $O(k\log^2(k/\delta))$ centers. `MUNSC` operates by executing several instances of a new center-selection procedure, called `SelectProc`, where each instance is applied to a stream prefix of a different scale. `SelectProc` decides which centers to select using a novel technique, in which a small stream prefix is used to estimate a truncated version of the optimal cost of the entire stream. This truncated version ignores outliers, and so can be estimated reliably. The value of the estimate is used to determine which centers to select from the rest of the stream. The multiscale use of `SelectProc` by `MUNSC` allows `MUNSC` to select only a quasi-linear number of centers.

As part of our analysis, we propose a new concept of a *linear bin division*, which allows a high-probability association of the costs of stream subsets, while selecting a number of centers that is independent of the stream length. This improves a previous construction of Meyerson et al. [2004].

The guarantees for `MUNSC` are provided in our main theorem, Theorem 4.1, stated in Section 4. Table 1 below compares our guarantees to previous works; See Section 2 for additional details. We note that our algorithm and guarantees are easily extendable to k-means clustering, as well as to any other clustering objective that satisfies the weak triangle inequality. `MUNSC` uses as a black box an offline k-median approximation algorithm with a constant approximation factor. Whenever the offline algorithm is efficient [e.g., Charikar et al., 2002, Li and Svensson, 2016], then so is `MUNSC`.

Limitations. The proposed algorithm is the first constant-approximation no-substitution clustering algorithm that makes no structural assumptions on the input data set. The only limitation is the assumption that the input order is random. This assumption is useful in cases where there is no adversary and any order is as likely, as in the example applications mentioned above. Moreover, a random order is a standard assumption in many streaming settings, satisfied also whenever the input stream is drawn i.i.d. from a distribution. In some cases, the randomness assumption might hold only approximately. In these cases, we expect a graceful degradation of the guarantees. We leave for future work a comprehensive treatment of input orders that are neither random nor adversarial. We expect the techniques of the current work to serve as a cornerstone in such a treatment.

Paper structure. We discuss related work in Section 2. The formal setting and notations are defined in Section 3. The algorithm and its guarantees are given in Section 4. In Section 5, we outline the proof of the main result. Some parts of the proof are deferred to appendices, which can be found in the supplementary material. We conclude with a discussion in Section 6.

2 Related Work

Several works have studied settings related to the no-substitution k-median clustering setting. Table 1 summarizes the upper bounds mentioned below. First, some works studied related settings under an adversarial arrival order. Liberty et al. [2016] studied online k-means clustering, in which centers are sequentially selected, and each observed point is allocated to one of the previously selected centers. The proposed algorithm can be applied to the no-substitution setting, yielding an approximation factor of $O(\log(n))$, where n is the stream length. For this algorithm to select a sublinear number of centers, the aspect ratio of the input data must be bounded. Bhaskara and Rwanpathirana [2020] studied the same setting as Liberty et al. [2016], improving the approximation factor to a constant, under

Table 1: Comparing our guarantees to previous works. n is the stream length. Abbreviations: LSS16: Liberty et al. [2016], BR20: Bhaskara and Rwanpathirana [2020], BM20: Bhattacharjee and Moshkovitz [2021] Mo21: Moshkovitz [2021], HS20: Hess and Sabato [2020].

Reference	arrival order	assumptions	approximation factor	number of centers
LSS16	adversarial	bounded aspect ratio	$O(\log(n))$	$O(k\log^2(n))$
BR20	adversarial	bounded aspect ratio	constant	$O(k\log^2(n))$
BM20	adversarial	data properties	$O(k^3)$	$O(k\log(k)\log(n))$
Mo21	random	none	exponential in k	$O(k^5)$ or $O(k\log(n))$
HS20	random	bounded diameter	constant+additive	k
This work	random	none	constant	$O(k\log^2(k))$

the same assumptions. Bhattacharjee and Moshkovitz [2021] explicitly studied the no-substitution setting, and provided an approximation factor of $O(k^3)$, under a different assumption on the input data set.

The setting of a random arrival order has been studied in several recent works. Hess and Sabato [2020] proposed an algorithm that selects exactly k centers. They obtained a constant approximation factor, with an additional additive term that vanishes for large streams, under the assumption that the metric space has a bounded diameter. Their guarantee is with high probability. Moshkovitz [2021] obtained an approximation factor that is exponential in k, with a number of centers that is linear in k and logarithmic in the stream length n. Assuming a known stream length, the same work also obtained an approximation factor which is exponential in k while selecting $O(k^5)$ centers. Both guarantees hold only with a constant probability. As can be seen in Table 1, to date, the only known constant approximation algorithms for no-substitution k-median clustering impose structural assumptions on the input data.

A related sequential clustering setting is the streaming setting [e.g., Guha et al., 2000, Ailon et al., 2009, Chen, 2009, Ackermann et al., 2012, Braverman et al., 2016], in which the main restriction is the amount of memory available to the algorithm. This setting allows substituting selected centers, but algorithms in this setting can be used in the no-substitution setting, by collecting all the centers ever selected. However, we are not aware of any algorithm in this setting with a competitive bound on the total number of selected centers.

3 Setting and Notation

For an integer i, denote $[i] := \{1, \ldots, i\}$. Let (X, ρ) be a finite metric space, where X is a set of size n and $\rho : X \times X \to \mathbb{R}_+$ is a metric. For a point $x \in X$ and a set $T \subseteq X$, we denote $\rho(x, T) := \min_{y \in T} \rho(x, y)$. For an integer $k \geq 2$, a k-clustering of X is a set of (at most) k points from X which represent cluster centers. Throughout this work, whenever an item is selected based on minimizing ρ, we assume that ties are broken based on a fixed arbitrary ordering. Given a set $S \subseteq X$, the *k-median cost* of T on S is $\mathsf{cost}(S, T) := \sum_{x \in S} \rho(x, T)$. The *$k$-median clustering problem* aims to select a k-clustering T of X with a minimal overall cost $\mathsf{cost}(X, T)$. We denote by OPT an optimal solution to this problem: $\mathrm{OPT} \in \mathrm{argmin}_{T \subseteq X, |T| \leq k} \mathsf{cost}(X, T)$. In the *sequential no-substitution* k-median setting that we study, X is not known a priori. The points from X are presented to the algorithm sequentially, in a random order. We assume that the stream length, n, is provided as input to the algorithm; see Section 6 for a discussion on supporting an unknown stream length. The algorithm may select an observed point as a center only before observing the next point. Any selected point cannot later be removed or substituted. The goal of the algorithm is to select a small number of centers, such that with a high probability, the overall cost of the selected set on the entire X approximates the optimal k-median cost, $\mathsf{cost}(X, \mathrm{OPT})$. In other words, we seek an algorithm with a low competitive ratio, in the sense of competitive analysis [Borodin and El-Yaniv, 2005]. An *offline k-median algorithm* $\mathcal{A}$ is an algorithm that takes as input a finite set

Algorithm 1 The sequential clustering algorithm: MUNSC

input $\delta \in (0,1)$, $k \in \mathbb{N}$, $n \in \mathbb{N}$ (stream length), $\mathcal{A}$ (an offline k-median algorithm), sequential access to the input stream.

1: $\alpha_1 \leftarrow \delta/(4k)$; For $i \in \mathbb{N}$, $\alpha_i \leftarrow \alpha_1 \cdot 2^{i-1}$.
2: $I \leftarrow \lfloor \log_2(1/(6\alpha_1)) \rfloor$, $\delta' \leftarrow \delta/(I+1)$. $\triangle$ *I is set so that $\alpha_{I+1} \in (1/12, 1/6]$.*
3: Prepare $I+1$ copies of SelectProc, indexed by $i \in [I+1]$, as follows:
  In all copies, use inputs $k, n, \mathcal{A}$, $\tau \leftarrow \phi_{\alpha_{I+1}}$, and set the confidence parameter to δ'.
  In copy $i \in [I+1]$, set $\alpha \leftarrow \alpha_i$.
  In all but the last copy, set $M \leftarrow 1$ and $\gamma \leftarrow 2\alpha_i$.
  In the last copy (index $I+1$), set $M \leftarrow \log(8k_+/\delta')$, and $\gamma \leftarrow 1 - 2\alpha_{I+1}$.
4: Read each point from the input stream and feed it to each of the copies of SelectProc. If any of the copies selected the point, then select it as a center.

of points S and the parameter k, and outputs a k-clustering of S, denoted $\mathcal{A}(S,k)$. For $\beta \geq 1$, we say that $\mathcal{A}$ is a *β-approximation* offline k-median algorithm on (X,ρ), if for all input sets $S \subseteq X$, $\mathsf{cost}(S, \mathcal{A}(S,k)) \leq \beta \cdot \mathsf{cost}(S, \mathrm{OPT}_S)$, where OPT_S is an optimal solution with centers from S. Formally, $\mathrm{OPT}_S \in \mathrm{argmin}_{T \subseteq S, |T| \leq k} \mathsf{cost}(S,T)$.

4 The algorithm

In this section, we describe the proposed algorithm, MUNSC. MUNSC receives as input the value of k, the confidence level $\delta \in (0,1)$ and the total stream length n. We further assume access to some black-box offline k-median algorithm $\mathcal{A}$. The guarantees of MUNSC depend on the approximation factor guaranteed by $\mathcal{A}$. MUNSC has only sequential access to the input stream. Our main result is the following theorem, showing that MUNSC obtains a constant approximation factor with a quasi-linear number of centers. A proof sketch of the theorem is given in Section 5. The full proof is provided in the supplementary.

Theorem 4.1. *Let $k, n \in \mathbb{N}$. Let $\delta \in (0,1)$. Let (X,ρ) be a metric space of size n. Let T_{out} be the set of centers selected by MUNSC for the input parameters δ, k, n. Suppose that the input stream is a random permutation of X, and that the input black-box algorithm $\mathcal{A}$ is a β-approximation offline k-median algorithm. Then, with a probability at least $1-\delta$,*

1. $|T_{\mathrm{out}}| = O\big(k \log^2(k/\delta)\big)$, *and*
2. $\mathsf{cost}(X, T_{\mathrm{out}}) \leq C\beta \cdot \mathsf{cost}(X, \mathrm{OPT})$, *where $C > 0$ is a universal constant.*

We first give, in Section 4.1, a short overview of the structure of MUNSC. A core element of the algorithm is the internal procedure SelectProc, which is used by MUNSC with several different sets of input parameters, creating a multiscale effect that we discuss below. We explain the details of SelectProc in Section 4.2. Then, in Section 4.3, we explain the design of MUNSC.

4.1 Algorithm structure overview

MUNSC is listed in Alg. 1. It works as follows: It initializes $\Theta(\log(k/\delta))$ copies of the procedure SelectProc, each with a different set of input parameters, where we define $\phi_\alpha := 150 \log(32k/\delta)/\alpha$ for $\alpha > 0$. It then iteratively reads the points from the input stream one by one, and feeds each point to each of the copies of SelectProc, so that each copy sequentially observes the entire input stream. Whenever any copy of SelectProc selects a point as a center, this point is selected by MUNSC as a center.

The internal procedure SelectProc uses a robust cost estimation approach to select a small number of centers. The procedure gets as input several technical parameters, in addition to the input parameters of MUNSC. We explain the procedure and the meaning of the technical parameters in Section 4.2. The values of the technical parameters set by MUNSC for each of the copies of SelectProc ensure that MUNSC select a small number of centers, while obtaining a constant approximation factor. This is done using a multiscale approach. We discuss this in detail in Section 4.3.

Algorithm 2 Internal procedure: `SelectProc`

input $\delta \in (0,1)$, $k \in \mathbb{N}$, $n \in \mathbb{N}$, $\mathcal{A}$ (an offline k-median algorithm),
sequential access to the input stream.
Technical parameters: $\alpha \in (0, \frac{1}{6}]$, $\gamma \in (\alpha, 1-2\alpha]$, $M \in \mathbb{N}$, $\tau > 0$.

$\triangle$ ***Phase 1*** *(Calculate an initial clustering T_α)*:
$P_1 \leftarrow$ the first αn points from the input stream.
$T_\alpha := \{c_1, \ldots c_{k_+}\} \leftarrow \mathcal{A}(P_1, k_+)$ $\triangle$ *Only calculation, no centers are actually selected here.*
$\triangle$ ***Phase 2*** *(Estimate the cost of T_α on large optimal clusters)*:
$P_2 \leftarrow$ next αn points from the input stream.
$\psi \leftarrow \frac{1}{3\alpha}\mathsf{cost}_{2\alpha(k+1)\phi_\alpha}(P_2, T_\alpha)$ $\triangle$ *ψ estimates the cost of T_α on large optimal clusters.*
$\triangle$ ***Phase 3*** *(Select centers)*:
$\forall i \in [k_+],\ n_i \leftarrow 0$, $\texttt{Near}_i \leftarrow \texttt{FALSE}$. $\triangle$ *n_i counts selected points associated with c_i.*
$\texttt{Near}_i$ indicates if a point close to c_i was selected.
for γn iterations **do**
  Read the next point x from the input stream.
  $i \leftarrow \mathrm{argmin}_{i \in [k_+]}\, \rho(x, c_i)$. $\triangle$ *Find the closet center to x in T_α.*
  if $\rho(x, c_i) > \frac{\psi}{k\tau}$ or $\neg\texttt{Near}_i$ or $n_i \leq M$ **then**
    Select x as a center. $\triangle$ *This is the only line in which actual selections occur.*
    $n_i \leftarrow n_i + 1$.
    If $\rho(x, c_i) \leq \frac{\psi}{k\tau}$ **then** $\texttt{Near}_i \leftarrow \texttt{TRUE}$.
  end if
end for

4.2 The internal procedure: `SelectProc`

In this section, we present the internal procedure, `SelectProc`. It has been observed in previous work [Liberty et al., 2016] that knowing in advance the optimal cost on the entire stream allows a successful center selection. Under a random arrival order, one may hope that obtaining a good estimate for the optimal cost would be straight-forward. However, since the metric space is unbounded, even a small number of outliers can bias the cost estimate considerably. We overcome this challenge by showing that it suffices to estimate a version of the optimal cost that ignores outliers, and that this version can be estimated reliably from a small prefix of the stream. Our analysis is based on a distinction between small and large optimal clusters, which we formally define in Section 5 below.

`SelectProc`, listed in Alg. 2, uses the following notation. For two sets $S, T \subseteq X$ and an integer r, denote by $\mathrm{far}_r(S,T) \subseteq S$ the set of r points in S that are the furthest from T according to the metric ρ. If $|S| < r$, we define $\mathrm{far}_r(S,T) := S$ and call $\mathrm{far}_r(S,T)$ a *trivial far set*. Denote by $\mathsf{cost}_r(S,T) := \mathsf{cost}(S \setminus \mathrm{far}_r(S,T), T)$ the cost of T on S after discounting the r points that incur the most cost. Let $k_+ := k + 38\log(32k/\delta)$. `SelectProc` receives the same input parameters as `MUNSC`. In addition, it requires several technical parameters, denoted α, γ, M and τ. We explain the meaning of these parameters in the proper context below.

`SelectProc` works in three phases. For $i \in [3]$, we denote by P_i the set of points that are read in phase i. In each of the first two phases, an α fraction of the points in the stream is read. In the last phase, a γ fraction of the points in the stream is read. Note that depending on the values of α and γ, the procedure may ignore a suffix of the stream. The first two phases are used for calculations. Centers are selected only during the third phase.

In the first phase, a reference clustering T_α is calculated using the input offline algorithm $\mathcal{A}$. Note that the centers in T_α cannot be selected as centers by `SelectProc`, since $\mathcal{A}$ calculates T_α only after observing all the points in the phase. In the second phase, an estimate ψ of the cost of T_α is calculated. In the third phase, `SelectProc` observes points from the stream one by one, deciding for each one whether it should be selected as a center. For each observed point, `SelectProc` first finds the center $c_i \in T_\alpha$ which is closest to it. A point is then selected as a center if it satisfies one of three conditions: (1) its distance from c_i is more than $\psi/(k\tau)$ (where τ is one of the input technical parameters); (2) c_i does not yet have M associated points (where M is one of the technical parameters); or (3) no point close to c_i has been selected so far, as maintained by the Boolean variable $\texttt{Near}_i$. In Theorem 5.1 in Section 5, we provide a guarantee on the output of `SelectProc`, which upper bounds the number of

centers selected by SelectProc and the cost obtained by these centers, as a function of the technical parameters.

Challenges and solutions in SelectProc. We give here an informal explanation of the workings of SelectProc. The full analysis is provided in Section 5 and the supplementary. Consider the clusters induced by some fixed optimal k-median clustering on X. Call these clusters *optimal clusters*. Optimal clusters that are large (that is, include a sufficiently large fraction of X) are easy to identify from a small random subset of points. Therefore, approximate centers for these clusters will be identified in the first phase of SelectProc by the black box algorithm $\mathcal{A}$, and will be included in T_α. Thus, the cost of T_α on the subset of points that belong to large optimal clusters will be close to optimal. Note that T_α is a clustering with $k_+ > k$ centers. This overcomes the fact that the data set might include outliers which are not proportionally represented in each phase. Searching for a k-clustering in P_1 might thus lead to a clustering that is too biased towards such outliers. By increasing the size of the solution searched in the first phase to k_+, this allows the solution to include the outliers as centers, while still choosing also centers that are close to the optimal centers of all large clusters. The selection conditions of the third phase further guarantee that at least M points are selected near each center $c_i \in T_\alpha$ which represents a large cluster, thus making sure that a center which is very close to each such c_i is selected. This allows bounding the obtained cost on points in large optimal clusters.

An additional challenge is to ensure that points in small optimal clusters are not too far from the set T_{out} of all selected centers. Since the metric space is unbounded, even a single such point can destroy the constant approximation factor. To overcome this, SelectProc selects a point near each center in T_α, as well as all the points that are far from T_α. The threshold $\psi/(k\tau)$, which defines a point as far, is set so that the number of points selected from large optimal clusters can be bounded. This bound crucially relies on the accuracy of ψ as an estimate for the cost of T_α on large clusters. The required accuracy is obtained by ignoring the points that are furthest from T_α when calculating the cost on P_2 (see line 6). For small clusters, they hold a small number of points by definition, thus the number of such points selected as centers is also bounded.

4.3 The clustering algorithm: MUNSC

As described above, MUNSC runs several copies of SelectProc on the same input stream: each point is read from the input stream and then fed to each of the copies of SelectProc. Each of these copies selects some centers, and the set of all selected centers is the solution of MUNSC. The difference between the copies is in the value of the technical input parameters. First, the value of α (the stream fraction for the first and second phases) is progressively doubled, starting with $\alpha_1 = \delta/(4k)$ and ending with $\alpha_{I+1} = \alpha_1 \cdot 2^I$, where I is set so that $\alpha_{I+1} \in (1/12, 1/6]$. The value of τ (which controls the selection threshold) is set to $\phi_{\alpha_{I+1}}$ in all copies, based on the largest value of α. In all but the last copy, γ (the stream fraction for the third phase) is set to $2\alpha_i$, where i is the copy index. This means that the first and second phase of copy i are each of size α_i, while the third phase is of size $2\alpha_i$ (the rest of the stream is ignored by copy i). Therefore, copy i observes a $4\alpha_i = 2\alpha_{i+1}$ fraction of the stream. Thus, the first two phases of copy $i+1$ exactly overlap with the fraction of the stream observed by copy i. Figure 1 illustrates the overlap of phases in consecutive copies of SelectProc. In the last copy, indexed by $I+1$, γ is set to $1 - 2\alpha_{I+1}$, thus this copy reads the entire stream. It can be seen that each point, except for the first $2n\alpha_1$ points, participates in the third phase of exactly one copy of SelectProc. As our analysis below shows, this overlap between the phases guarantees that the set of centers selected by the copies of SelectProc obtains a small cost on all the small optimal clusters. The last copy is slightly different: in addition to setting the length of the third phase γ to include all the points of the stream, it also sets the technical parameter M to a number larger than 1. We show below that in this way, the selected set of centers obtains a small cost on large optimal clusters as well.

The use of several copies of SelectProc with different phase sizes ensures that MUNSC selects only a quasi-linear number of centers, by overcoming the following challenge: On the one hand, the first phase and the second phase must be small, otherwise we might miss a point that needs to be selected, since no points are selected in these two phases. On the other hand, small first and second phases lead to a poor quality of the reference clustering T_α. The multiscale approach makes sure that almost all points participate in some selection phase, while at the same time improving the quality of T_α as

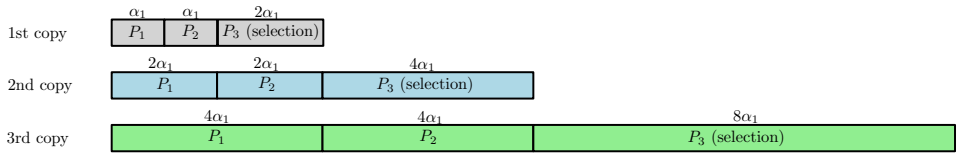

Figure 1: Illustrating the phases in copies $1, 2, 3$ of `SelectProc` within `MUNSC`.

α grows. Our analysis below shows that in this way, the number of centers selected by each copy remains similar, even though larger values of α lead to a larger selection phase.

Using `MUNSC` in a bounded memory setting. `MUNSC` can be used in a bounded memory setting, by using a 1-pass streaming clustering algorithm as the black-box algorithm $\mathcal{A}$. For instance, the algorithm of Charikar et al. [2003] requires $O(k \cdot \mathrm{polylog}(n))$ memory. `MUNSC` can be implemented to require additional memory of only $O(k \log(k/\delta))$ on top of the requirements of the black-box algorithm. We now explain in full how this can be done. First, consider the memory requirements of `SelectProc`: In phase 1, the points can be fed to A one by one and do not need to be saved in memory. Only T_α, the $k_+ = O(k + \log(\frac{1}{\delta}))$ output points of $\mathcal{A}$, need to be saved in memory. In phase 2, calculating the truncated cost of T_α on P_2 can be done by saving at most $2\alpha \cdot (k+1)\phi_\alpha = O(k \log(\frac{k}{\delta}))$ at any time, so as to identify the farthest points from T_α, which are the ones to disregard. In phase 3, the points are inspected one by one and do not need to be saved in memory (if one wishes to save the selected centers in memory, although this is not in principle required in our setting, then an additional $O(k \log(\frac{k}{\delta}))$ factor is added). In total, `SelectProc` can be implemented using a memory size of $O(k \log(\frac{k}{\delta}))$ + the memory requirments of $\mathcal{A}$. `MUNSC` runs $O(\log(\frac{k}{\delta}))$ parallel runs of `SelectProc`. Thus, it requires at most a memory size of $O(\log(\frac{k}{\delta})(k \log(\frac{k}{\delta})$ + the memory requirements of $\mathcal{A}))$. In fact, this can be made even more memory-efficient, by using an algorithm $\mathcal{A}$ which is a streaming anytime algorithm. In this case, it is possible to use a single copy of $\mathcal{A}$ for all the parallel runs of `SelectProc`, since P_1 in the i'th copy of `SelectProc` is the prefix of P_1 in the $i+1$'th copy (see Figure 1). That is, $\mathcal{A}$ is run on the entire stream and whenever a phase 1 of any copy ends, its current set of centers is retrieved for this copy. In this case, the memory size required by `MUNSC` is $O(k \log^2(\frac{k}{\delta}))$ + the memory requirements of $\mathcal{A}$.

5 Analysis

In this section, we give an overview of the proof of Theorem 4.1. The supplementary provides the full proof. Denote the centers in the optimal solution OPT by $\{c_i^*\}_{i \in k}$, and the clusters induced by OPT (the *optimal clusters*) by $\{C_i^*\}_{i \in k}$. Formally, $C_i^* := \{x \in X \mid i = \mathrm{argmin}_{j \in [k]} \rho(c_j^*, x)\}$. Optimal clusters with fewer than ϕ_α points are α-*small* optimal clusters, and the complement are α-*large*. Denote the indices of α-small and α-large clusters by $I^\alpha_{\mathrm{small}} := \{i \in [k] \mid |C_i^*| < \phi_\alpha\}$ and $I^\alpha_{\mathrm{large}} := [k] \setminus I^\alpha_{\mathrm{small}}$, and the points in these clusters by $C^\alpha_{\mathrm{small}} := \bigcup_{i \in I^\alpha_{\mathrm{small}}} C_i^*$ and $C^\alpha_{\mathrm{large}} := \bigcup_{i \in I^\alpha_{\mathrm{large}}} C_i^*$. To prove Theorem 4.1, we provide the following guarantee on the centers selected by `SelectProc`.

Theorem 5.1. *Let $k, n \in \mathbb{N}$. Let $\delta \in (0,1)$. Let (X, ρ) be a metric space of size n. Let T_{out} be the set of centers selected by `SelectProc` for the input parameters $\delta, k, n, \alpha \in (0, 1/6]$, $\gamma \in (\alpha, 1-2\alpha]$, $M \in \mathbb{N}$, $\tau > 0$. Suppose that the input stream is a random permutation of X, and that $\mathcal{A}$ is a β-approximation offline k-median algorithm. Then, with a probability at least $1 - \delta/2$,*

1. $|T_{\mathrm{out}}| = O\left(\frac{\gamma}{\alpha} k \log(k/\delta) + k\tau + (k + \log(k/\delta))M\right)$;
2. *For any $i \in [k]$, if $c_i^* \in P_3$, then*

$$\mathsf{cost}(C_i^*, T_{\mathrm{out}}) \leq \mathsf{cost}(C_i^*, \{c_i^*\}) + \frac{|C_i^*|}{k\tau}(36\beta + 20)\mathsf{cost}(X, \mathrm{OPT}); \tag{1}$$

3. *If $\gamma = 1 - 2\alpha$, $\tau = \phi_\alpha$ and $M = \log(8k_+/\delta)$, then*

$$\mathsf{cost}(C^\alpha_{\mathrm{large}}, T_{\mathrm{out}}) \leq \mathsf{cost}(C^\alpha_{\mathrm{large}}, \mathrm{OPT}) + (468\beta + 260)\mathsf{cost}(X, \mathrm{OPT}). \tag{2}$$

The proof of Theorem 4.1, provided in Appendix A, uses part 1 to bound the total number of centers, and parts 2 and 3 to bound the cost of small and large optimal clusters, respectively. The statement of Theorem 5.1 hints to the reason for the multiscale design of `MUNSC`: On the one hand, part 2 gives an approximation upper bound only if the cluster center appears in the third phase. For this to hold with a high probability, the third phase needs to be very large. On the other hand, for a quasi-linear bound on selected centers in part 1, the ratio between the third phase and the first phases (γ/α) needs to be small. The multiscale design overcomes this by keeping the ratio constant in each copy of `SelectProc`, while ensuring that the union of all third phases is almost the entire stream.

In the rest of the section, we give the proof of Theorem 5.1: In Section 5.1, we introduce the concept of *linear bin divisions*, and derive its properties. In Section 5.2, we give an overview of the proof of Theorem 5.1, using the results of Section 5.1. The full proof is provided in the supplementary.

5.1 Linear bin divisions

A main tool in the proof of Theorem 5.1 is partitioning sets of points from the input stream into subsets, such that each of the subsets is probably well represented in a relevant random subset of the stream. Meyerson et al. [2004] defined the concept of a *bin division* of X with respect to a clustering T, which is a partition of X into bins of equal size, where points are allocated to bins based on their distance from T. Here, we define the concept of a *linear bin division*, in which the bins linearly increase in size. This gradual increase allows keeping the size of the first bin independent of the stream length, while still proving that with a high probability, the overlap of each of the bins in the division with random subsets of the stream is close to expected. The fixed size of the first bin is crucial for deriving guarantees that are independent of the stream length. In addition, the ratio between adjacent bin sizes is kept bounded, which allows proving a bounded approximation factor.

Definition 5.2 (Linear bin division)**.** *Let $W, T \subseteq X$ be finite sets, and $z \in \mathbb{N}$. A* z-linear bin division of W with respect to T *is a partition $\mathcal{B} = (\mathcal{B}(1), \ldots, \mathcal{B}(L))$ of W (for an integer L) such that:*

1. *If $z \leq |W|$, then $\forall i \in [L], |\mathcal{B}(i)| \geq z \cdot (i+1)/2$. Otherwise, the bin division is called* trivial*, and defined as $\mathcal{B} := \mathcal{B}(1) := W$.*
2. *$|\mathcal{B}(1)| \leq \frac{5}{2} z$.*
3. *$\forall i \in [L-1], |\mathcal{B}(i+1)|/|\mathcal{B}(i)| \leq 3/2$.*
4. *$\forall i \in [L-1]$, and $\forall x \in \mathcal{B}(i), x' \in \mathcal{B}(i+1)$, it holds that $\rho(x, T) \geq \rho(x', T)$.*

A linear bin division exists for any size of W: For $|W| \leq z$, the conditions trivially hold. For $|W| \geq z$, the three first properties hold for the following allocation of bin sizes: Let L be the largest integer such that $B := \sum_{i \in [L]} z \cdot (i+1)/2 \leq |W|$. Set the size of $\mathcal{B}(i)$ to $z \cdot (i+1)/2 + (|W| - B)/L$. Property 1 clearly holds. Property 2 holds since $(|W| - B)/L \leq (z(L+2)/2)/L \leq 3z/2$. Property 3 holds since $(i+2+a)/(i+1+a) \leq 3/2$ for all $i \geq 1$ and any non-negative a. To satisfy property 4, allocate the elements of W into the bins in descending order of their distance from T.

We say that a set is *well-represented* in another set if the size of its overlap with the set is similar to expected. This extends naturally to bin divisions. Formally, this is defined as follows.

Definition 5.3 (Well-represented)**.** *Let W be a finite set and $A, B \subseteq W$. We say that B is* well-represented in A for W *if $|B \cap A|/|B| \in [r/2, (3/2)r]$, where $r := |A|/|W|$. We say that a linear bin division $\mathcal{B}$ of W is well represented in A for W if each bin in $\mathcal{B}$ is well represented in A for W.*

The following lemma shows that if A is selected uniformly at random from W, then any fixed set B is well-represented in A with a high probability. Moreover, the same holds for any z-linear bin division of W with a sufficiently large z. The proof of the lemma is provided in Appendix B.

Lemma 5.4. *Let W be a finite set. Let $B \subseteq W$ be a set, and let $\mathcal{B}$ be a z-linear bin division of some subset of W with respect to some T, for some integer z. Let $A \subseteq W$ be a set of size $r|W|$ selected uniformly at random from W. Then the following hold:*

1. *With a probability at least $1 - 2e^{-\frac{r|B|}{10}}$, B is well-represented in A for W.*
2. *If $z \geq 10 \log(4/\delta)/r$, then with a probability $1 - \delta$, $\mathcal{B}$ is well represented in A for W.*

We now state and prove a main property of linear bin divisions, which will allow us to use sub-streams of the input stream to bound the cost of a clustering on X.

Lemma 5.5. *Let $W \subseteq X$ and let $z \in \mathbb{N}$. Let $\mathcal{B}$ be a z-linear bin division of W with respect to some T. Let $A \subseteq W$ and $r \in (0,1)$. If $\forall i \in [L], |\mathcal{B}(i) \cap A|/|\mathcal{B}(i)| \leq r$, then $\mathsf{cost}(A \setminus \mathcal{B}(1), T) \leq \frac{3}{2} r \mathsf{cost}(W, T)$.*

Proof. Let L be the number of bins in $\mathcal{B}$. We first prove the following inequality, which relates the cost of the intersection of a bin with A to the cost of the preceding bin.

$$\forall i \in [L-1],\ \mathsf{cost}(A \cap \mathcal{B}(i+1), T) \leq \frac{3}{2} r \cdot \mathsf{cost}(\mathcal{B}(i), T). \tag{3}$$

To prove Eq. (3), fix $i \in [L-1]$, and denote $b := \max_{x \in \mathcal{B}(i+1)} \rho(x, T)$. By the assumptions, we have $|A \cap \mathcal{B}(i+1)| \leq r|\mathcal{B}(i+1)|$. Hence, $\mathsf{cost}(A \cap \mathcal{B}(i+1), T) \leq r|\mathcal{B}(i+1)| \cdot b$. By property 3 of linear bin divisions, $|\mathcal{B}(i+1)| \leq \frac{3}{2}|\mathcal{B}(i)|$. Therefore, $\mathsf{cost}(A \cap \mathcal{B}(i+1), T) \leq \frac{3}{2} r|\mathcal{B}(i)| \cdot b$. In addition, by property 4 of linear bin divisions and the definition of b, $\forall x \in \mathcal{B}(i), b \leq \rho(x, T)$. Therefore, $\mathsf{cost}(\mathcal{B}(i), T) \geq |\mathcal{B}(i)| \cdot b$. Combining the two inequalities, we get Eq. (3). It follows that:

$$\mathsf{cost}(A \setminus \mathcal{B}(1), T) = \sum_{i=2}^{L} \mathsf{cost}(A \cap \mathcal{B}(i), T) \leq \frac{3}{2} r \sum_{i=1}^{L-1} \mathsf{cost}(\mathcal{B}(i), T) \leq \frac{3}{2} r \mathsf{cost}(W, T).$$

This proves the statement of the lemma. □

To prove Theorem 5.1, we define an event, denoted E, which holds with a high probability. We prove each of the claims in the theorem under the assumption that this event holds. To define the event, we first provide some necessary notation. Consider a run of `SelectProc` with some fixed set of input parameters, and assume that the input stream is a random permutation of the points in X. Recall that $T_\alpha = \{c_1, \ldots, c_{k_+}\}$ is the clustering calculated in the first phase of `SelectProc`. Denote the clusters induced by T_α on X by $C_1, \ldots, C_{k_+}$, where $C_i := \{x \in X \mid i = \operatorname{argmin}_{j \in k_+} \rho(c_j, x)\}$. We define several sets and bin divisions, which will be used to define the required event. Let $\mathcal{B}_a$ be a $(\phi_\alpha/15)$-linear bin division of X with respect to OPT. Define $\mathrm{F}_b := \mathrm{far}_{k\phi_\alpha}(X \setminus P_1, T_\alpha)$, and let $\mathcal{B}_b$ be a $(\phi_\alpha/3)$-linear bin division of $X \setminus (P_1 \cup \mathrm{F}_b)$ with respect to T_α. Define $\mathrm{F}_c := \mathrm{far}_{4(k+1)\phi_\alpha}(X \setminus P_1, T_\alpha)$, and let $\mathcal{B}_c$ be a $(\phi_\alpha/3)$-linear bin division of $X \setminus (P_1 \cup \mathrm{F}_c)$ with respect to T_α. Lastly, define $\mathrm{F}_d := \mathrm{far}_{5(k+1)\phi_\alpha}(X \setminus P_1, T_\alpha)$. Note that each of these objects may be trivial if the stream is small.

The event E is defined as the following conjunction: (1) For each $i \in I^\alpha_{\mathrm{large}}$, C^*_i is well-represented in P_1 and in $P_1 \cup P_2$ for X. (2) Each of F_b and F_c is trivial or well-represented in P_2 for $X \setminus P_1$; F_d is trivial or well-represented in P_3 for $X \setminus P_1$. (3) $\mathcal{B}_a$ is trivial or well-represented in P_1 for X; Each of $\mathcal{B}_b$ and $\mathcal{B}_c$ is trivial or well-represented in P_2 for $X \setminus P_1$, and (4) For each $i \in [k_+]$, one of the first $\log_2(8k_+/\delta)$ points observed from C_i in P_3 is closer to c_i than at least half of the points in $C_i \cap P_3$.

The following lemma, proved in Appendix C, shows that E holds with a high probability.

Lemma 5.6. *Consider a run of `SelectProc` with fixed input parameters. Assume that the input stream is a random permutation of the points of X. Then E holds with a probability at least $1 - \delta/2$.*

5.2 Theorem 5.1: proof overview

In this section, we give an overview of the proof of Theorem 5.1. The full proof is given in Appendix D. In the first phase of `SelectProc`, the clustering T_α is calculated using the offline algorithm $\mathcal{A}$. We first bound the cost of T_α on points in α-large optimal clusters.

Lemma 5.7. *If E holds, then $\mathsf{cost}(C^\alpha_{\mathrm{large}}, T_\alpha) \leq (18\beta + 10)\mathsf{cost}(X, \mathrm{OPT})$.*

In the second phase of `SelectProc`, the goal is to estimate a truncated version of the cost of T_α on X. This truncated version ignores the cost on the outliers, which are the $k\phi_\alpha$ furthest points from T_α in X. Setting the estimate to $\psi = \frac{1}{3\alpha}\mathsf{cost}_{2\alpha(k+1)\phi_\alpha}(P_2, T_\alpha)$ ignores the furthest $2\alpha(k+1)\phi_\alpha$ points in P_2, which with a high probability include all the $k\phi_\alpha$ outliers in X. The following lemma bounds ψ between two versions of a truncated cost of T_α.

Lemma 5.8. *If E holds, then $\frac{1}{9}\mathsf{cost}_{5(k+1)\phi_\alpha}(X \setminus P_1, T_\alpha) \leq \psi \leq \mathsf{cost}_{k\phi_\alpha}(X, T_\alpha)$.*

The proof of this lemma uses the linear bin divisions studied in Section 5.1. In particular, the upper bound uses Lemma 5.5, and the lower bound uses the specific construction of the bin sizes.

From the lemmas above, it can be seen that ψ is also upper bounded by the optimal cost, as follows. α-small optimal clusters are smaller than ϕ_α, thus their total size is less than $k\phi_\alpha$. It follows that

$$\mathsf{cost}_{k\phi_\alpha}(X, T_\alpha) \leq \mathsf{cost}(X \setminus C^\alpha_{\text{small}}, T_\alpha) = \mathsf{cost}(C^\alpha_{\text{large}}, T_\alpha). \tag{4}$$

Combining Lemma 5.7, Eq. (4) and Lemma 5.8, we get that ψ is upper bounded by the optimal cost.

In the third phase, the points selected as centers include (among others) all the points with a distance of more than $\psi/(k\tau)$ from T_α. This allows bounding the cost of T_{out} on points in some of the optimal clusters by a linear expression in $\psi/(k\tau)$. Combined with the upper bound on ψ, this allows proving part 2 of Theorem 5.1. Part 3 bounds the cost of T_{out} on large optimal clusters, for specific settings of the technical parameters of `SelectProc`. To prove this part, we show that under these parameter settings, the centers selected based on the reference clustering T_α include sufficiently many points around each large optimal cluster. We then show that one of these points is close to the optimal center.

Lastly, part 1 of Theorem 5.1 upper bounds the number of centers selected by `SelectProc`. The following lemma upper-bounds the number of points that are selected because they are far from T_α.

Lemma 5.9. *Let* $N := |\{x \in P_3 \mid \rho(x, T_\alpha) > \psi/(k\tau)\}|$. *Under E,* $N \leq 8(k+1)\phi_\alpha \frac{\gamma}{1-\alpha} + 9k\tau$.

The proof Lemma 5.9 uses the lower bound on ψ shown in Lemma 5.8, to conclude that there cannot be too many points that are more than $\psi/(k\tau)$ far from T_α, since this would make the truncated cost in the lower bound larger than ψ. We note that the resulting upper bound is independent of the stream size, due to the specific construction of the bin sizes in the linear bin division. Using this lemma, we can prove part 1 of `SelectProc`. The full proof of Theorem 5.1 is given in Appendix D.

6 Discussion

In this work, we provided `MUNSC`, the first constant-approximation algorithm for sequential no-substitution k-median clustering that does not require structural assumptions on the input data. Moreover, the number of centers selected by `MUNSC` is only quasi-linear in k.

`MUNSC` can also be used when the stream length n is not known in advance, using a simple doubling trick. Moshkovitz [2021] showed that an unknown stream length with an approximation factor that does not depend on n require a logarithmic dependence on n in the number of selected centers. Indeed, a doubling trick for `MUNSC` would preserve the approximation factor, increasing the number of selected centers only by the unavoidable $\Theta(\log(n))$ factor.

Funding Transparency Statement

Sivan Sabato and Tom Hess were supported in part by the Israel Science Foundation (grant No. 555/15) and by the United-States-Israel Binational Science Foundation (BSF) (grant no. 2017641). Michal Moshkovitz has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation program (grant agreement No. 882396), by the Israel Science Foundation (grant number 993/17), Tel Aviv University Center for AI and Data Science (TAD), and the Yandex Initiative for Machine Learning at Tel Aviv University.

References

Marcel R Ackermann, Marcus Märtens, Christoph Raupach, Kamil Swierkot, Christiane Lammersen, and Christian Sohler. StreamKM++: A clustering algorithm for data streams. *Journal of Experimental Algorithmics (JEA)*, 17:2–4, 2012.

Charu C Aggarwal. A framework for diagnosing changes in evolving data streams. In *Proceedings of the 2003 ACM SIGMOD international conference on Management of data*, pages 575–586. ACM, 2003.

Nir Ailon, Ragesh Jaiswal, and Claire Monteleoni. Streaming k-means approximation. In *Advances in neural information processing systems*, pages 10–18, 2009.

Aditya Bhaskara and Aravinda Kanchana Rwanpathirana. Robust algorithms for online k-means clustering. In *Algorithmic Learning Theory*, pages 148–173, 2020.

Robi Bhattacharjee and Michal Moshkovitz. No-substitution k-means clustering with adversarial order. In *Algorithmic Learning Theory*, pages 345–366. PMLR, 2021.

Allan Borodin and Ran El-Yaniv. *Online computation and competitive analysis*. cambridge university press, 2005.

Vladimir Braverman, Dan Feldman, and Harry Lang. New frameworks for offline and streaming coreset constructions. *arXiv preprint arXiv:1612.00889*, 2016.

Moses Charikar, Sudipto Guha, Éva Tardos, and David B Shmoys. A constant-factor approximation algorithm for the k-median problem. *Journal of Computer and System Sciences*, 65(1):129–149, 2002.

Moses Charikar, Liadan O'Callaghan, and Rina Panigrahy. Better streaming algorithms for clustering problems. In *Proceedings of the thirty-fifth annual ACM symposium on Theory of computing*, pages 30–39. ACM, 2003.

Ke Chen. On coresets for k-median and k-means clustering in metric and euclidean spaces and their applications. *SIAM Journal on Computing*, 39(3):923–947, 2009.

Devdatt P Dubhashi and Desh Ranjan. Balls and bins: A study in negative dependence. *BRICS Report Series*, 3(25), 1996.

Sudipto Guha, Nina Mishra, Rajeev Motwani, and Liadan O'Callaghan. Clustering data streams. In *Foundations of computer science, 2000. proceedings. 41st annual symposium on*, pages 359–366. IEEE, 2000.

Tom Hess and Sivan Sabato. Sequential no-substitution k-median-clustering. In *International Conference on Artificial Intelligence and Statistics*, pages 962–972. PMLR, 2020.

Kumar Joag-Dev and Frank Proschan. Negative association of random variables with applications. *The Annals of Statistics*, pages 286–295, 1983.

Kingsly Leung and Christopher Leckie. Unsupervised anomaly detection in network intrusion detection using clusters. In *Proceedings of the Twenty-eighth Australasian conference on Computer Science-Volume 38*, pages 333–342. Australian Computer Society, Inc., 2005.

Shi Li and Ola Svensson. Approximating k-median via pseudo-approximation. *SIAM Journal on Computing*, 45(2):530–547, 2016.

Edo Liberty, Ram Sriharsha, and Maxim Sviridenko. An algorithm for online k-means clustering. In *2016 Proceedings of the Eighteenth Workshop on Algorithm Engineering and Experiments (ALENEX)*, pages 81–89. SIAM, 2016.

Adam Meyerson, Liadan O'callaghan, and Serge Plotkin. A k-median algorithm with running time independent of data size. *Machine Learning*, 56(1-3):61–87, 2004.

Michal Moshkovitz. Unexpected effects of online no-substitution k-means clustering. In *Algorithmic Learning Theory*, pages 892–930. PMLR, 2021.

Olfa Nasraoui, Jeff Cerwinske, Carlos Rojas, and Fabio Gonzalez. Performance of recommendation systems in dynamic streaming environments. In *Proceedings of the 2007 SIAM International Conference on Data Mining*, pages 569–574. SIAM, 2007.

Andriy Shepitsen, Jonathan Gemmell, Bamshad Mobasher, and Robin Burke. Personalized recommendation in social tagging systems using hierarchical clustering. In *Proceedings of the 2008 ACM conference on Recommender systems*, pages 259–266. ACM, 2008.

Bichen Zheng, Sang Won Yoon, and Sarah S Lam. Breast cancer diagnosis based on feature extraction using a hybrid of k-means and support vector machine algorithms. *Expert Systems with Applications*, 41(4):1476–1482, 2014.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See **limitations** in Section 1.
   (c) Did you discuss any potential negative societal impacts of your work? [No] We developed a generic clustering algorithm. We do not see a direct path to negative applications.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes]
   (b) Did you include complete proofs of all theoretical results? [Yes] All proofs have been provided in full. Some of the proofs have been deferred to the supplementary, due to lack of space.
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [N/A]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [N/A]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [N/A]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [N/A]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [N/A]
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant costs, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]