Diverse Sampling of Streaming Data

by

Aizana Turmukhametova

B.S. Mathematics, and Computer Science and Engineering, M.I.T., 2012

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Engineering in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2013

Author . . . .
Department of Electrical Engineering and Computer Science
August 23, 2013

Certified by . . . .
Piotr Indyk
Professor
Thesis Supervisor

Certified by . . . .
Samuel Madden
Professor
Thesis Supervisor

Accepted by . . . .
Professor Albert R. Meyer
Chairman, Masters of Engineering Thesis Committee

Diverse Sampling of Streaming Data

by

Aizana Turmukhametova

Submitted to the Department of Electrical Engineering and Computer Science
on August 23, 2013, in partial fulfillment of the
requirements for the degree of
Master of Engineering in Electrical Engineering and Computer Science

Abstract

This thesis addresses the problem of diverse sampling as a dispersion problem and proposes solutions that are optimized for large streaming data. Finding the optimal solution to the dispersion problem is NP-hard. Therefore, existing and proposed solutions are approximation algorithms. This work evaluates the performance of different algorithms in practice and compares them to the theoretical guarantees.

Thesis Supervisor: Piotr Indyk
Title: Professor

Thesis Supervisor: Samuel Madden
Title: Professor

Acknowledgments

First of all, I would like to thank Sam Madden and Piotr Indyk for being my mentors. I feel honored to have been working with you. I am very grateful for your time, support and guidance.

I would like to thank everyone in the DB group for making my time at CSAIL enjoyable. I would like to thank Adam Marcus for the discussions about the class project. I would like to thank Kelly Casteel, my class project partner, for working with me. I would like to thank Todd Mostak for providing the Twitter data set.

I would like to thank the E.E.C.S. undergraduate office, especially Anne Hunter and Linda Sullivan, for doing excellent work and being very supportive and friendly.

Last, but not least, I would like to thank my family and friends for always being there for me. I am grateful to my parents and my sister for all their care and encouragement. I am especially grateful to my boyfriend, Kuat, for all the love and support he has given me.

Contents

List of Figures

List of Tables

Chapter 1

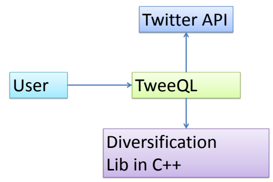
Introduction

1.1 Motivation

Streaming data of a large volume with high content similarity is increasingly common nowadays. For example, Twitter users produce a large volume of tweets every day and many of those tweets are very similar in content. Since the volume of the data set is too large, a sample of a smaller size is desired. Random sampling will resemble the overall distribution of tweets and is likely to contain some redundancy. For a user, a more diverse set of tweets might be more informative about an event or a topic of interest. In such a case, one might wish to select a small but diverse sample from the data set. Diversity is of interest in any system where items that are too similar to each other make the quality of the answer set worse. This thesis formulates diverse sampling as a dispersion problem, studies the existing solutions and offers new solutions that are optimized for streaming data model.

The initial motivation for this thesis came from a class project for Database Systems course where my project partner and I implemented *diverse selection* operators for TweeQL [15, 16]. TweeQL is a stream query processing language that allows users to use SQL-style queries over Twitter API. It provides `select`, `filter`, `group by` and `aggregate` operators as well as user-defined functions. We added a new operator `diverse`, which given a number

Figure 1-1: The system overview of the TweeQL extension project.



k, limits the answer to the most diverse subset of size k it can find. Figure 1-1 shows the overview of the class project. Users send SQL-style queries to TweeQL, which processes the queries and pulls tweets using Twitter API. In case of a *diverse* operator, it uses the diversification module written in C++ that solves the most diverse subset problem. While my project partner worked on extending the TweeQL with the new operator, I worked on the diversification module that models diversification as a dispersion problem and offers multiple algorithmic solutions. This thesis builds on the work I did on the project and does more in-depth analysis of different algorithms and presents new solutions.

1.2 Thesis Outline

The next chapter gives the formal definition of the problem. Chapter 3 discusses the related work. Chapter 4 presents the algorithms and provides a theoretical comparison of them. Chapter 5 presents an experimental comparison of the algorithms and Chapter 6 concludes.

Chapter 2

Formal Definition

Suppose we are given a data set D of n items, with items arriving one at a time, in a "stream". We do not know in advance the value of n. For a given $k \in \mathbb{N}$, we would like to select a subset $S \subset D$ of size k, such that S is as diverse as possible. In order to do that, we must have a well defined definition of diversity. Section 2.1 presents distance functions for measuring diversity. Given such distance functions, Section 2.2 defines what it means for a subset to be the most diverse.

2.1 Measure of Diversity

To evaluate diversity, we need to have a well defined **distance function**

$$d : D \times D \to \mathbb{R}$$

that measures the diversity between any two pairs of items. An edge is defined as a pair of items with a corresponding distance between them. The greater the value of the distance, the greater the diversity between the items. Ideally, the distance function should be a metric distance. A metric distance must satisfy the following conditions for all $x, y, z \in D$:

1. non-negativity: $d(x, y) \geq 0$

2. identity of indiscernibles: $d(x, y) = 0$ if and only if $x = y$

3. symmetry: $d(x, y) = d(y, x)$

4. triangle inequality: $d(x, y) \leq d(x, z) + d(z, y)$

A distance function that violates the triangle inequality might still be of a practical interest as the most diverse subset will be well defined. However, for such an arbitrary function, an efficient (polynomial) algorithm with an approximation guarantee might not exist unless $P = NP$ [18].

Given a distance function, we can define the diversity of a set S in two ways:

1. $minEdge(S)$ which equals to the minimum distance over all pairs in S, i.e.

$$minEdge(S) = \min_{x \neq y \in S} d(x, y)$$

Intuitively, the set is as diverse as the least diverse pair it contains.

2. $avg(S)$ which equals to the average distance over all pairs in S, i.e.

$$avg(S) = \frac{1}{|S|(|S| - 1)} \sum_{x, y \in S} d(x, y)$$

Intuitively, the set is as diverse as the average diversity of all pairs it contains.

In general, any distance function that can capture the desired diversity can be used. In the context of Twitter data, we can specify the following distance functions for measuring diversity.

2.1.1 Geographic Diversity

Given latitudes and longitudes of geocoded items such as tweets, we can use the **geographic distance** between locations as a measure of diversity. For geographic distances,

$minEdge(S)$ is commonly used to evaluate the diversity of a set. This kind of diversity is useful when one wants to see samples from all over the world, not just the US and Europe that produce majority of the tweets.

2.1.2 Text Content Diversity

Given a set of documents, like tweets, we can use **angular distance**, or *arccos*-distance, between high dimensional word vectors as a measure of text content diversity. One can also use **Jaccard distance**, or 1 - Jaccard index of word sets, but it will not satisfy the triangle inequality. Text diversity is useful when one wants to see the variety of topics, opinions and messages, not just the most popular ones.

2.1.3 User Diversity

To have a good user diversity distance, one has to be able to measure how similar two users are. For example, the similarity can be based on the followers graph in a system like Twitter. Since we do not have information about the twitter users, as an experiment, we decided to use a simple binary similarity. If two users are the same, then the distance is 0; if two users are different the distance is 1.

2.2 k-Diverse Subsets

Given diversity measures of the previous section 2.1 and a positive integer k, one can define the **dispersion problem** as to find subset $S \subset D$ of size k, such that the diversity, $minEdge(S)$ or $avg(S)$, is maximized. I call such problems k-diverse subset problems as follows:

Definition 1. *Given a data set D and diversity distance function d, a **k-diverse maxmin subset** is $S \subset D$ of size k, such that $minEdge(S)$ is maximized over all possible subsets.*

Definition 2. *Given a data set D and diversity distance function d, a **k-diverse maxavg subset** is $S \subset D$ of size k, such that $avg(S)$ is maximized over all possible subsets.*

Both of these problems are known to be NP-hard, and finding an *absolute*, or constant additive error, approximation is also NP-hard [18]. Hence, this thesis explores *relative*, or constant factor, approximation algorithms. The next chapter 3 describes the related work and known results about these problems.

Chapter 3

Related Work

3.1 Diversity

Diverse answers are useful for recommendation systems, search engines, and other systems that have to select a small set of items from a large corpus of data. The proposed formulation of diversity is similar to some recommendation systems [1, 23]. A survey on search result diversification mentions dispersion as one of the approaches to diversity [8]. Similar to related works in [1, 7, 9, 20], this thesis formulates diversity as a dispersion problem and adapts static algorithms proposed in [18].

Diversifying answers has been studied in the context of search engines. The main goal of a search engines is to get the most relevant and high quality items at the top of the results. Search engines use probabilistic models for defining the problem and evaluating results. Such models can be extended to take diversity into account [3, 21]. Since the main goal is different from ours, such search diversity models are unsuitable for the purposes of this thesis.

In the context of databases, diverse sampling might be desired to achieve an approximate answer to a query. [13] provides a survey of different sampling techniques and algorithms in streaming data environments. Among the surveyed techniques, congressional sampling can be used to obtain diverse samples. It is useful for approximating `group by` queries

[2]. Such sampling can be useful for achieving a username diversity, but not suitable for a multi-dimensional diversity. For example, uniqueness of latitudes and longitudes does not guarantee that the points are well dispersed on the map. This is the main difference with other related work on diverse query results in [19, 2].

3.2 Dispersion Problem

The dispersion problem is an old problem originally formulated in the context of locating facilities on a network, such that either the minimum distance or the average distance between facilities is maximized [10, 18, 5]. In operations research, it is often called **p-dispersion** problem [10]. In theory of graphs, maximizing the average distance is called **heaviest k-subgraph** problem [4]. It can also be formulated as an Integer Linear Program [14]. There are also different varieties of dispersion problems. [5] provides a good survey of dispersion problems with different objective functions.

Both versions of the dispersion problem, maximizing the minimum distance and the average distance, are known to be NP-hard [18]. In general, if the distance function does not satisfy the triangle inequality, an approximation can also be NP-hard [18]. If the distance function satisfies the triangle inequality, reasonable approximation algorithms can be found. [10] gives a good overview of different heuristics for finding an approximation to k-diverse maxmin subset problem.

As a baseline comparison, I use static greedy algorithms that have been proposed in [18]. `MaxMinStatic` guarantees a factor of 2-approximation to the k-diverse maxmin subset problem, and `MaxAvgStatic` guarantees a factor of 4-approximation to the k-diverse maxavg subset problem. Unlike such existing solutions, in this thesis, I propose to use simple greedy algorithms that are optimized for the streaming data. My algorithms, `MaxMinGreedy` and `MaxAvgGreedy`, do not give any theoretical guarantees, but in practice get results that are similar to the proposed baseline algorithms.

The authors of [17] independently developed similar streaming algorithms. Their results are similar to mine in showing that greedy streaming approach does well in practice. I extend this work with a more efficient streaming greedy maxmin algorithm, which has $O(k \log k)$ insertion time unlike $O(k^2)$ proposed in [17].

In addition, in this thesis I propose `Doubling` algorithm as an approximation to a k-diverse maxmin subset problem. The algorithm was originally used for approximating a **k-center problem** in a streaming environment [6]. k-center is a clustering problem of selecting k cluster centers from the given set of n items. Its objective is to minimize the maximum radius, when each point is assigned to the closest center. It is also known to be an NP-hard problem [6]. [12] gives a good overview on the theory and practice of clustering data streams. Note that the `MaxMinStatic` algorithm proposed in [18] is similar to the 2-approximation algorithm for the k-center problem in [11]. Despite the connection between k-center and k-diverse maxmin subset problems, it is possible to construct examples where an optimal solution to the k-center problem is arbitrarily bad at approximating k-diverse subset. Therefore, adopting algorithms for k-center to solve k-diverse subset problem is not trivial.

Chapter 4

Algorithms

This chapter describes in detail all the algorithms that have been implemented for the experimentation purposes. The algorithms are separated into two types: static and streaming. I refer to solutions as static if they do not take into account the streaming nature of data and store all data points in order to calculate the final answer at the end. Thus, they require $O(n)$ space. Streaming solutions, on the other hand, process each item as it arrives and maintain the best solution. They require much less space, namely $O(k)$ or $O(k^2)$.

4.1 Static Solutions

4.1.1 MaxMinStatic

`MaxMinStatic` starts with selecting a *maximum edge* from all edges in the data set D, and then keeps adding a new item to the result set S until k items have been selected. When choosing an item to add, it maximizes $minEdge(S)$. As a result, it guarantees a 2-approximation to the k-diverse maxmin subset problem [18]. The pseudo code is shown as Algorithm 1.

`MaxMinStatic` takes $O(n^2)$ time to find the maximum edge and takes $O(kn)$ time to find S. In the streaming model, it might be best to keep track of the *maximum edge* as the

Algorithm 1 MaxMinStatic

Find $x_i, x_j \in D$, such that $d(x_i, x_j)$ is maximized
$S \leftarrow \{x_i, x_j\}$
while $|S| < k$ **do**
    Find $x \in D \backslash S$ such that $minEdge(S \cup x)$ is maximum
    $S \leftarrow S \cup \{x\}$
end while
Output S

points arrive, resulting in $O(n)$ per item processing time and $O(kn)$ final answer calculation time.

4.1.2 MaxAvgStatic

`MaxAvgStatic` is very similar to `MaxMinStatic`. It also starts with selecting a *maximum edge* and then keeps updating the result set S until k items have been selected. However, unlike `MaxMinStatic`, it chooses an item to maximize $avg(S)$. Therefore, it results in a 4-approximation solution to the k-diverse maxavg subset problem. The pseudo code is shown as Algorithm 2. It takes $O(n^2)$ time to find the *maximum edge* and $O(kn)$ time to find S. One can implement it to have $O(n)$ per item processing time and $O(kn)$ final answer calculation time.

Algorithm 2 MaxAvgStatic

Find $x_i, x_j \in D$, such that $d(x_i, x_j)$ is maximized
$S \leftarrow \{x_i, x_j\}$
while $|S| < k$ **do**
    Find $x \in D \backslash S$ such that $avg(S \cup x)$ is maximum
    $S \leftarrow S \cup \{x\}$
end while
Output S

4.2 Streaming Solutions

4.2.1 Random

The `Random` algorithm selects a subset S of size k uniformly at random from the data set D. It does not have any objectives such as to maximize $minEdge(S)$ or $avg(S)$. It serves as a baseline solution that is the most efficient. **Reservoir sampling** can be used to efficiently choose a random subset from the streaming data [22]. It takes constant time to process each item and constant time to "calculate" the final answer.

4.2.2 MaxMinGreedy

`MaxMinGreedy` is a greedy streaming algorithm that keeps track of the current "best" solution S to k-diverse maxmin subset. When a new item x arrives, it examines whether swapping some item $y \in S$ with x increases the objective value $minEdge(S)$. Let the objective value after swapping $y \in S$ with x be

$$swap(x, y) = minEdge(\{x\} \cup S \backslash \{y\})$$

If swapping increases the objective value, then x is swapped with an item y that maximizes $swap(x, y)$. Otherwise, point x is discarded. The pseudo code for processing and item is shown as algorithm 3

Algorithm 3 MaxMinGreedy::insert(x)

```
if |S| < k then
    Add x to S
else
    if max_{y∈S}(swap(x, y)) > minEdge(S) then
        S ← {x} ∪ S\{y}
        where y is such that maximizes swap(x, y)
    end if
end if
```

It is possible to implement `MaxMinGreedy` with $O(k)$ space and $O(k^2)$ running time as in [17], or with $O(k^2)$ space and $O(k \log k)$ running time. To achieve $O(k \log k)$ running time, one should keep a balanced binary search tree of incident edges for each point in S.

4.2.3 MaxAvgGreedy

`MaxAvgGreedy` is a greedy streaming algorithm that keeps track of the current "best" solution S to k-diverse maxavg subset. Similar to `MaxMinGreedy`, it greedily swaps $y \in S$ with x, as to maximize the objective value. In this case, the objective value after swapping $y \in S$ with x is

$$avg(\{x\} \cup S \backslash \{y\})$$

Item x is discarded if swapping cannot increase the objective value. The pseudo code for `MaxAvgGreedy` item processing is shown as algorithm 4

Algorithm 4 MaxAvgGreedy::insert(x)

---

if $|S| < k$ **then**
  Add x to S
else
  find $y \in S$ that maximizes $avg(\{x\} \cup S \backslash \{y\})$
  if $avg(\{x\} \cup S \backslash \{y\}) > avg(S)$ **then**
    $S \leftarrow \{x\} \cup S \backslash \{y\}$
  end if
end if

It is possible to implement `MaxAvgGreedy` with $O(k)$ space and $O(k)$ running time [17].

4.2.4 Doubling Algorithm

The `Doubling` algorithm is a streaming algorithm that guarantees 8-approximation to k-diverse maxmin subset. The original idea was used to solve k-center clustering problem [6]. I have adopted it with a minor modification. It is shown as Algorithm 5 and Theorem 1 shows the proof the approximation bound.

Algorithm 5 Doubling::insert(x)

```
if |S| < k and |S*| = 0 then
    add x to S
    r ← minEdge(S)
else if ∃y ∈ S, such that d(x, y) ≤ 2r then
    ignore x
else if |S| < k then
    add x to S
else
    S* ← S
    add x to S
    r ← max(2r, minEdge(S))
    while ∃y, z ∈ S such that d(y, z) ≤ r do
        remove y from S
    end while
end if
```

Algorithm 6 Doubling::answer()

```
if |S| < k and |S*| = k then
    output S*
else
    output S
end if
```

The algorithm maintains set S of center points and value $r \in \mathbb{R}$ such that

Invariant 1. *r is lower bound on the minimum edge between items in S, i.e.*

$$minEdge(S) \geq r$$

Invariant 2. *For any point x that the algorithm has processed, $\exists y \in S$, such that*

$$d(x, y) \leq 2r$$

i.e. x is within $2r$ of some center $y \in S$.

The initial value of S are the first k points and $r = minEdge(S)$. As we process new points, we want these invariants to hold, so that we can make a guarantee on the approximation factor. We want S to be of size k. When a new point arrives and it cannot be discarded because of the invariant 2, we add it to S. If S becomes of size $k + 1$, $|S|$ is reduced by doubling r or increasing it as much as necessary, and removing the centers in S that violate the invariant 1. Unlike in [6] where r is doubled until the size of S can be reduced, I assign the new radius r to $max(2r, minEdge(S))$. The reduction in the size of S might result in $|S| < k$. Since we expect more points to arrive, we will allow the size of S to be smaller than k. We can state this as another invariant.

Invariant 3. *The size of S is less than or equal to k, i.e. $|S| \leq k$.*

When $|S| < k$ and the algorithm has to return the answer, it returns the previous set of size k saved as S^* as shown in algorithm 6.

It is not hard to see that the invariants hold after each insertion. Given the invariants, let's show that the algorithm achieves an 8-approximation to k-diverse maxmin subset.

Theorem 1. *Doubling algorithm achieves an 8-approximation to k-diverse maxmin subset.*

Proof. Suppose S_{opt} is the best solution to k-diverse maxmin subset. For any subset $S \subset D$, we have that

$$minEdge(S_{OPT}) \geq minEdge(S)$$

We have to show that

$$minEdge(S_{OPT}) \leq 8 \cdot minEdge(S)$$

Consider the three different cases in the answer that the algorithm gives:

1. The algorithm returns S of size $|S| < k$ because fewer than k items have been inserted.

2. The algorithm returns S of size k.

3. The algorithm returns S^* of size k because $|S| < k$.

Case 1 is trivial and case 2 is easy given the invariants 1 and 2. In case 2, we have that

$$r \leq minEdge(S)$$

Suppose $\exists x, y \in S_{OPT}$, such that they are within $2r$ of some center $z \in S$, then

$$minEdge(S_{OPT}) \leq d(x, y) \leq d(x, z) + d(z, y) \leq 4r \leq 4 \cdot minEdge(S)$$

If no two points in S_{OPT} are within $2r$ of some center $z \in S$, then there is one-to-one correspondence between centers S and items in S_{OPT}, such that the distance is less than or equal to $2r$ according to the invariant 2. In this case, consider the pair $x, y \in S$ that corresponds to $minEdge(S)$. Each of x and y have corresponding points $x', y' \in S_{OPT}$, such that

$$d(x, x') \leq 2r$$

$$d(y, y') \leq 2r$$

We get that

$$minEdge(S_{OPT}) \leq d(x', y') \leq d(x', x) + d(x, y) + d(y, y') \leq$$

$$\leq 4r + minEdge(S) \leq 5 \cdot minEdge(S)$$

This shows that case 2 gives us a factor of 5 approximation.

Now, let's prove case 3. We have that $|S| < k$. By the pigeonhole principle, $\exists x, y \in S_{OPT}$ such that x and y are within $2r$ of some center $z \in S$, i.e.

$$d(x, z) \leq 2r$$

$$d(y, z) \leq 2r$$

This gives us that

$$minEdge(S_{OPT}) \leq d(x, y) \leq 4r$$

Let r^* be the old value of r that corresponds to the solution S^* and let x be the point whose insertion caused the increase in r. There are two possibilities:

(i) $r = 2r^*$

(ii) $r = minEdge(S^* \cup \{x\})$

In case (i), we get that

$$minEdge(S_{OPT}) \leq 4r = 8r^* \leq 8 \cdot minEdge(S^*)$$

because S^* with r^* satisfied the invariant 1 before the insertion of x.

In case (ii), we have that

$$minEdge(S_{OPT}) \leq 4r = 4 \cdot minEdge(S^* \cup \{x\}) \leq 4 \cdot minEdge(S^*)$$

because adding x to S^* cannot increase the minimum edge value between the items.

This concludes the proof that the `Doubling` algorithm achieves 8-approximation to the k-diverse maxmin subset. □

It is possible to implement the `Doubling` algorithm with $O(k)$ space and $O(k^2)$ running time, or with $O(k^2)$ space and amortized $O(k \log k)$ running time as shown in [6].

4.3 Theoretical Comparisons

I have implemented the static and streaming algorithms described in this chapter. Table 4.1 lists all algorithms and summarizes their space and item processing time requirements.

Table 4.1: Implemented Algorithms

Algorithm	Insert	Answer	Space	Maximizes	Approximation
`MaxMinStatic`	$O(n)$	$O(kn)$	$O(n)$	min edge	2
`MaxAvgStatic`	$O(n)$	$O(kn)$	$O(n)$	avg	4
`Random`	$O(1)$	$O(1)$	$O(k)$	-	-
`MaxMin`	$O(k \log k)$	$O(1)$	$O(k^2)$	min edge	-
`MaxAvg`	$O(k)$	$O(1)$	$O(k)$	avg	-
`Doubling`	$O(k \log k)$	$O(1)$	$O(k^2)$	min edge	8

Chapter 5

Experiments

This chapter evaluates and compares the quality of results achieved by the implemented algorithms. The evaluation data set contains about six million geo-tagged tweets collected on 24-25 December of 2012. Section 5.1 evaluates the diversity achieved on the Twitter data set, and section 5.2 evaluates the runtimes of the proposed algorithms.

5.1 Result Quality

This section evaluates the quality of diversity achieved by the proposed algorithms. Subsection 5.1.1 shows examples of different types of diversity on the Twitter data set in order to demonstrate the validity of the diversification model. Given the k-diverse subset model, subsection 5.1.2 evaluates the approximation factors achieved by different algorithms.

5.1.1 Tweet Diversity Examples

Geographic diversity

Consider the geographic diversity discussed in the subsection 2.1.1. Figures 5-1, 5-2, and 5-3 show the results of selecting $k = 20$ tweets from $n = 10000$ tweets using `Random`, `MaxMinGreedy`, and `MaxAvgGreedy` algorithms correspondingly. As we can see from the pictures, `MaxMinGreedy` results in points well dispersed on the world map. The `Random`

Figure 5-1: Geographic diversity: `Random` selected tweets.

Figure 5-2: Geographic diversity: `MaxMinGreedy` selected tweets.

---

algorithm tends to select points located in the US and Europe, where the majority of tweets are from. `MaxAvgGreedy` tends to select points in two clusters, where the clusters are far away from each other, in order to maximize the average edge.

Figure 5-3: Geographic diversity: `MaxAvgGreedy` selected tweets.

Table 5.1: Example of a geographic diversity for $n = 10000$ and $k = 20$

Algorithm	Maximizes	Avg (miles)	Min Edge (miles)
`Random`	-	3740.626	15.597
`MaxMinStatic`	min edge	6309.610	2034.272
`MaxMinGreedy`	min edge	6232.295	1911.953
`Doubling`	min edge	5948.964	819.676
`MaxAvgStatic`	avg	6518.224	64.022
`MaxAvgGreedy`	avg	6510.230	60.351

Table 5.1 shows the values of minimum and average edges for all algorithms. `MaxMinStatic` and `MaxAvgStatic` achieve values similar to `MaxMinGreedy` and `MaxAvgGreedy`. The `Doubling` algorithm achieves values smaller than `MaxMinGreedy`, but much better than the `Random` algorithm. In general, to achieve a good geographic diversity, it is best to maximize the minimum edge.

Text Diversity

Consider the content diversity discussed in the subsection 2.1.2. Figures 5-4 and 5-5 show the world clouds of $k = 50$ tweets selected from $n = 5000$ tweets using `Random` and

Figure 5-4: Text diversity: `Random` selected tweets.

Figure 5-5: Text diversity: `MaxMinGreedy` selected tweets.

`MaxMinGreedy` algorithms, and Jaccard index for measuring diversity. From figure 5-4 we can see that `Random` algorithm has selected many tweets related to Christmas as it was the most popular topic during the Christmas day of 2012. In contrast, tweets selected by `MaxMinGreedy`, in figure 5-5, have more uniform word distribution as it maximizes for the content diversity.

Table 5.2: Example of text diversity for $n = 5000$ and $k = 50$

Algorithm	Maximizes	Avg (miles)	Min Edge (miles)
`Random`	-	0.9897	0.3333
`MaxMinStatic`	min edge	1.0	1.0
`MaxMinGreedy`	min edge	1.0	1.0
`Doubling`	min edge	0.9882	0.6250
`MaxAvgStatic`	avg	1.0	1.0
`MaxAvgGreedy`	avg	1.0	1.0

Table 5.3: Distribution of usernames with `Random` selection

Number of users	Number of occurrences
953	1
20	2
1	3
1	4

Table 5.2 shows the values for minimum and average edges for all algorithms. All algorithms except `Random` and `Doubling` achieve a perfect solution as 1.0 is the maximum possible value for an edge. The `Doubling` algorithm achieves a better minimum edge value compare to `Random`.

User Diversity

Consider the user diversity discussed in the subsection 2.1.3. Table 5.3 shows the distribution of usernames when selecting $k = 1000$ tweets from $n = 10000$ tweets using `Random` algorithm. All other algorithms that maximize for some diversity, achieve a perfect solution, i.e. all usernames are unique and edge values are all equal to 1. Of course, there are much better ways to achieve username diversity. The purpose of this example is to demonstrate that the proposed algorithms can achieve a diversity of users.

5.1.2 Quality of Proposed Algorithms

As we can see from the examples of the previous section, it is reasonable to model diversification as a k-diverse subset problem. Given such a model, we can measure the quality of the result set S by the value of the objective function. In case of k-diverse maxmin subset, the objective is to maximize $minEdge(S)$, and for k-diverse maxavg subset, the objective is to maximize $avg(S)$. Note that finding the optimal solution is not practical. A brute-force search algorithm will have to look at $\binom{n}{k}$ possible subsets. Therefore, we evaluate the quality by comparing the proposed algorithms to each other. `Random` algorithm can serve as a good baseline because it is the most efficient one possible. `MaxMinStatic` and `MaxAvgStatic` serve as a good baseline to the streaming algorithms, because they have a theoretical guarantee on the approximation factor.

This section presents experiments that show that `MaxMinGreedy` and `MaxAvgGreedy` achieve diversity results very similar to `MaxMinStatic` and `MaxAvgStatic` correspondingly. The `Doubling` algorithm does worse than `MaxMinGreedy` in practice, but unlike `MaxMinGreedy` guarantees a factor of 8-approximation in the worst case. All diversification algorithms achieve significantly better results than `Random` algorithm. All examples are shown for the geographic diversity. Similar results can be obtained for other diversity metrics.

k-Diverse Maxmin Subset Algorithms

Let us consider algorithms that maximize the minimum edge. Figure 5-6 shows the minimum edge value for a geographic diversity with $n = 10000$ and varying values of k. We can see that `MaxMinGreedy` and `MaxMinStatic` are very similar to each other and better than `Doubling`. Varying k while keeping n constant shows that the selected tweets become less diverse as the subset size increases. If we adopt `MaxMinStatic` as a baseline algorithm and repeat the experiment multiple times, we get the results shown in the figure 5-7. It shows the average ratio of the minimum edge to the value obtained by `MaxMinStatic` algorithm as we vary k and keep n constant. If we keep k constant and vary n, we get results shown in the figure 5-8. From the figures, we can see that the ratio of `MaxMinGreedy` to `MaxMinStatic` is about 1, which means that these two algorithms achieve very similar objective values. We can also see that `MaxMinGreedy` tends to do slightly better when k is much smaller compare to n. However, such difference is insignificant.

Figure 5-6: Geographic Diversity: k vs. minimum edge for $n = 10000$.

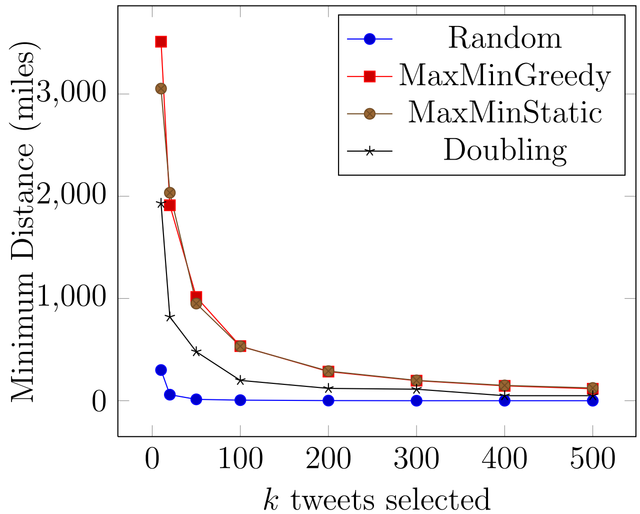
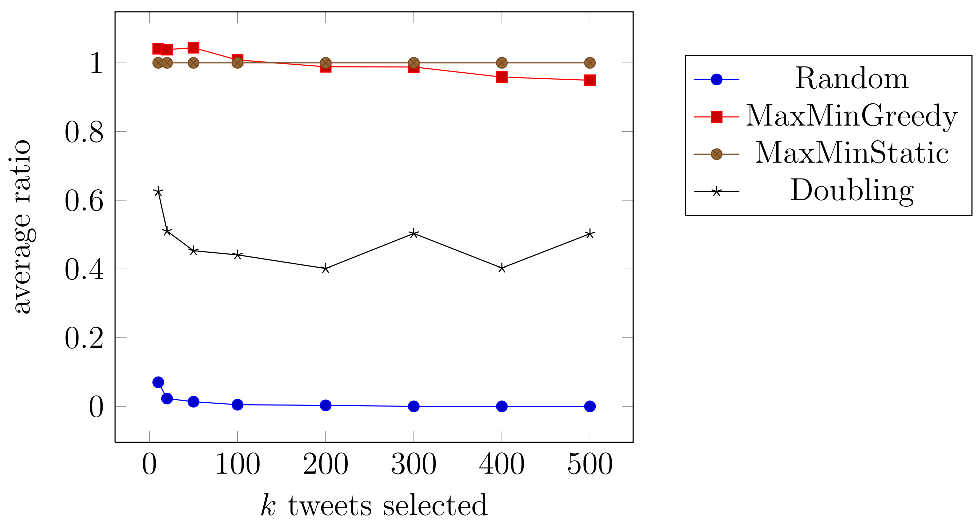
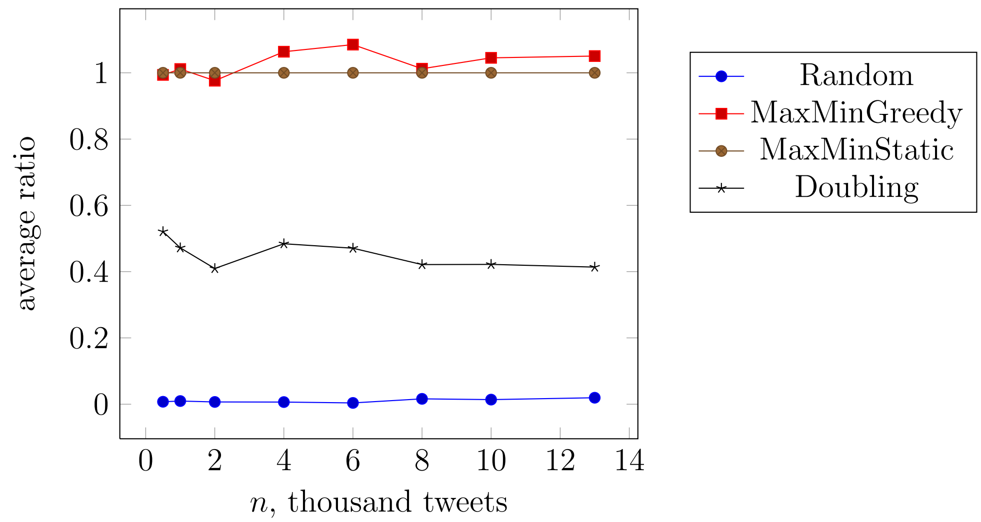


Figure 5-7: k vs. minimum edge relative to `MaxMinStatic` for $n = 10000$.



The `Doubling` algorithm achieves average ratios around 0.5, which means that it does two times worse than `MaxMinGreedy` and `MaxMinStatic`. The `Random` algorithm does the worst with an average ratio very close to 0.

Figure 5-8: n vs. minimum edge relative to `MaxMinStatic` for $k = 30$



---

k-Diverse Maxavg Subset Algorithms

`MaxAvgGreedy` and `MaxAvgStatic` maximize the average edge length.If we adopt `MaxAvgStatic` as a baseline algorithm and plot average ratios, we get the results shown in Figures 5-9 and 5-10. Figure 5-9 shows average ratios when keeping n constant and varying k, and figure 5-10 shows average ratios when keeping k constant and varying n. We can see that `MaxAvgGreedy` does slightly worse than `MaxAvgStatic`, but the ratio is very close to 1. The `Random` algorithm fluctuates more and achieves ratios around $0.7 - 0.8$. We can conclude that `MaxAvgGreedy` and `MaxAvgStatic` achieve similar values and do significantly better than a random algorithm.

5.2 Performance

This section shows the runtimes for the proposed algorithms on Intel(R) Pentium(R) 4, $2GB$, $3.8GHz$ machine. The implementation has been done using C++. For streaming algorithms we care about the insertion time of an item since they maintain current best answer at all times. For static algorithms, the insertion time can be as fast as saving an item or can be $O(n)$ if doing some processing upon insertion. Since we get the final answer only after

Figure 5-9: k vs. average edge relative to `MaxAvgStatic` for $n = 10000$.

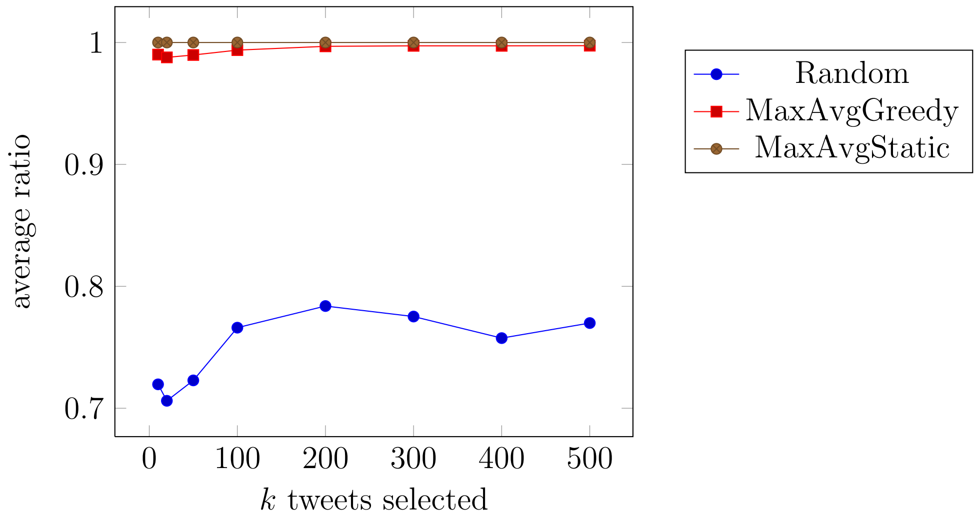
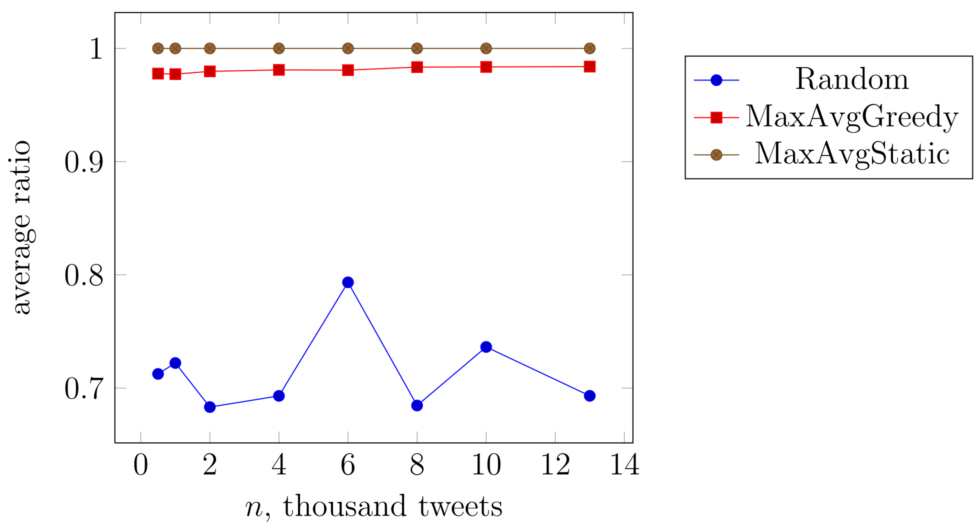


Figure 5-10: n vs. average edge relative to `MaxAvgStatic` for $k = 30$.



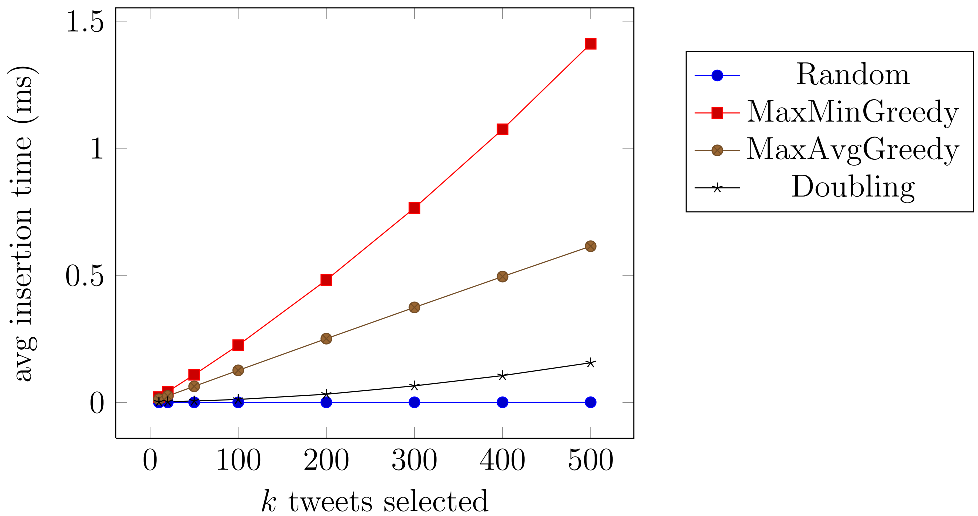
---

running the algorithm on all items, I report total runtimes for the static ones.

5.2.1 Insertion

Figure 5-11 shows insertion times for the streaming algorithms when varying k and keeping n constant. The insertion time of the `Random` algorithm stays constant because it is independent of k. The insertion time of `MaxAvgGreedy` grows linearly with k and the insertion time

Figure 5-11: k vs. average insertion time for $n = 10000$.



of `MaxMinGreedy` grows as $O(k \log k)$. The `Doubling` algorithm also grows as $O(k \log k)$, but for small values of k it is much faster than `MaxAvgGreedy` and `MaxMinGreedy`. This can be explained by the fact that the `Doubling` algorithm is more lazy and ignores points that are within $2r$ of some center point.

If we keep k constant and vary n, we get results shown in Figure 5-12. As expected, streaming algorithms do not depend on n and the insertion times stay constant as n increases.

5.2.2 Total Runtimes

If we compare the total runtimes of the proposed algorithms, then streaming algorithms are an order of magnitude faster than the static algorithms and scale much better for large data sets. Since the order of execution time is very different, I plot streaming and static algorithms separately. Figure 5-13 shows total runtimes for the streaming algorithms and figure 5-14 shows total runtimes for the static algorithms as we vary k and keep n constant. Total runtimes include item insertions and final answer calculation times. If we vary n and keep k constant, we get results shown in figures 5-16 and 5-15. `MaxAvgStatic` and `MaxMinStatic` scale quadratically with respect to n and have very similar execution times

Figure 5-12: n vs. average insertion time for $k = 30$.

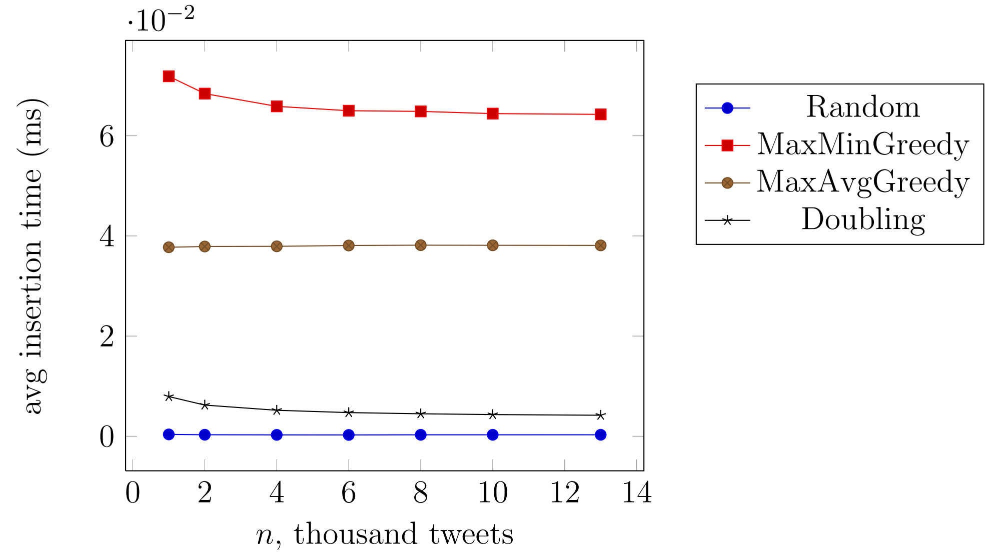
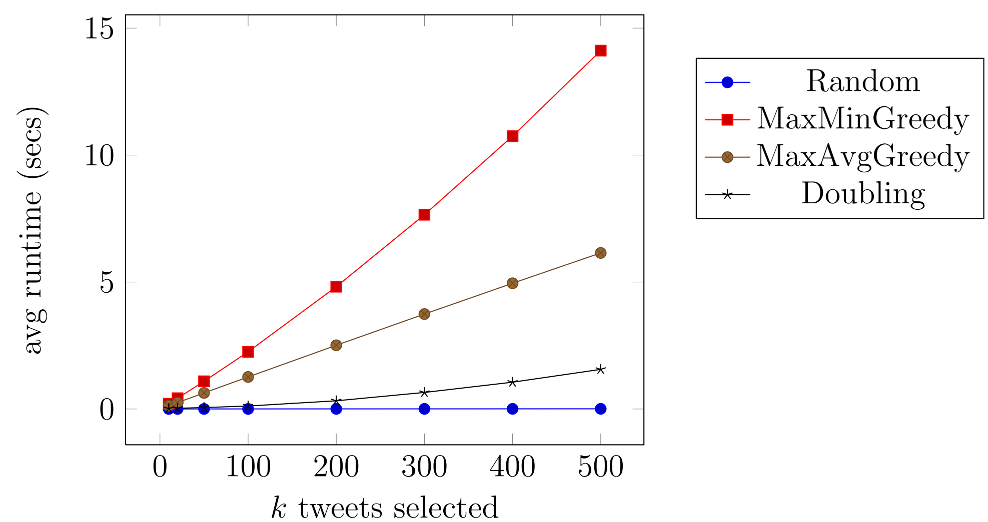


Figure 5-13: Streaming algorithms: k vs. average runtime for $n = 10000$.



---

as their complexity is similar. As expected, streaming algorithms have much better scaling for large n. The `Doubling` algorithm proves to be very efficient in practice compare to other streaming approaches.

Figure 5-14: Static algorithms: k vs. average runtime for $n = 10000$.

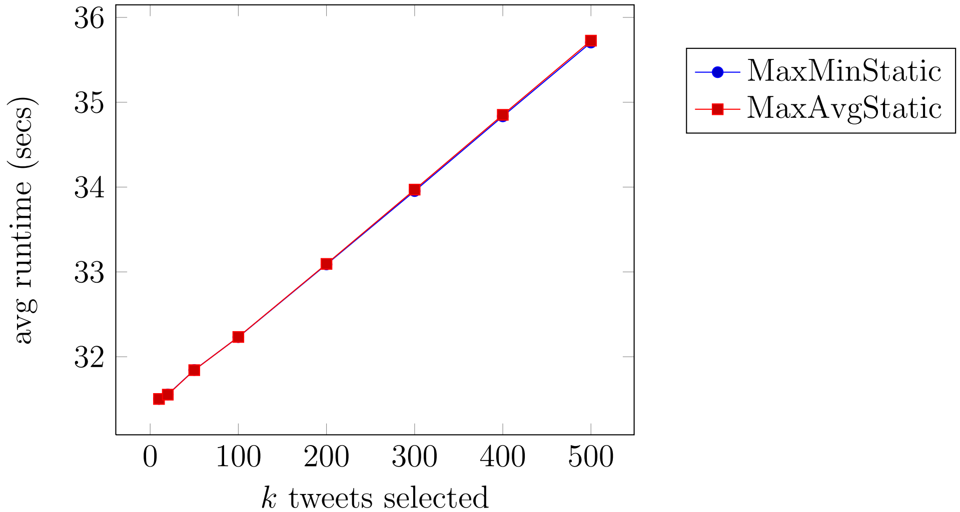


Figure 5-15: Streaming algorithms: n vs. average runtime for $k = 30$.

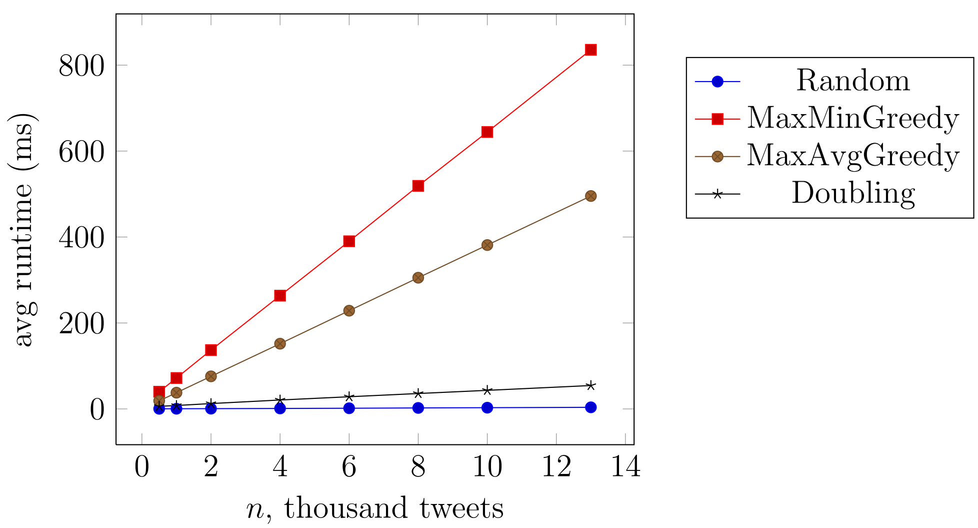
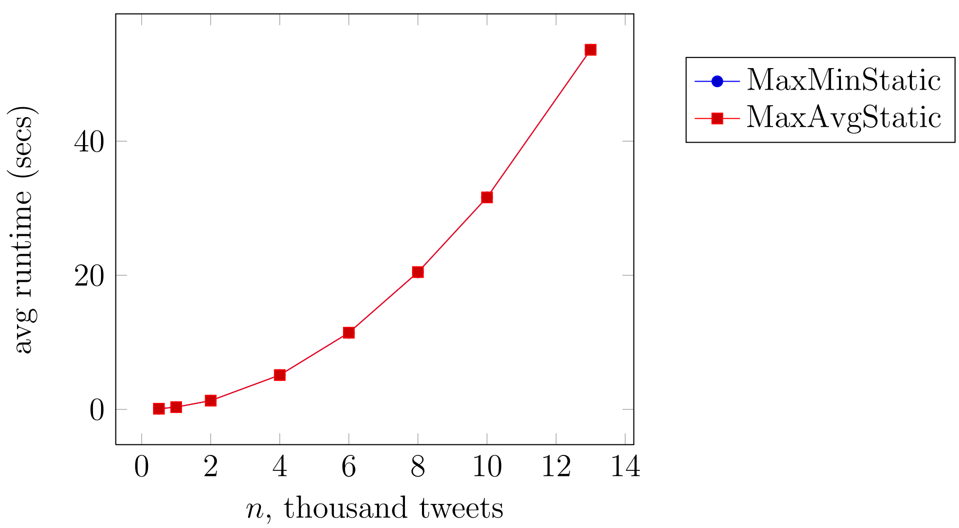


Figure 5-16: Static algorithms: n vs. average runtime for $k = 30$.



Chapter 6

Conclusion

In conclusion, the three major contributions of this thesis are:

1. The evaluation of proposed algorithms on the Twitter data set shows that formulating diversification as a dispersion problem is a reasonable approach and can be used to achieve different types of diversity.

2. Experiments show that `MaxMinGreedy` and `MaxAvgGreedy` achieve results that are very similar to `MaxMinStatic` and `MaxAvgStatic`, while being much more efficient both in streaming and non-streaming environment.

3. Theorem 1 proves that `Doubling` algorithm achieves an 8-approximation to k-diverse maxmin subset problem. Although in practice `Doubling` algorithm achieves worse result sets than `MaxMinGreedy`, it is more efficient and is recommended for large data sets. To my best knowledge, it is the first streaming algorithm that gives a constant factor guarantee on the quality of approximation.

Bibliography

[1] Sofiane Abbar, Sihem Amer-Yahia, Piotr Indyk, and Sepideh Mahabadi. Real-time recommendation of diverse related articles. In *Proceedings of the 22nd international conference on World Wide Web*, WWW '13, pages 1–12, Republic and Canton of Geneva, Switzerland, 2013. International World Wide Web Conferences Steering Committee.

[2] Swarup Acharya, Phillip B. Gibbons, and Viswanath Poosala. Congressional samples for approximate answering of group-by queries. *SIGMOD Rec.*, 29(2):487–498, May 2000.

[3] Albert Angel and Nick Koudas. Efficient diversity-aware search. In *Proceedings of the 2011 ACM SIGMOD International Conference on Management of data*, SIGMOD '11, pages 781–792, New York, NY, USA, 2011. ACM.

[4] Alain Billionnet. Different formulations for solving the heaviest k-subgraph problem. Technical report, 2002.

[5] Barun Chandra and Magns M. Halldrsson. Approximation algorithms for dispersion problems, 2001.

[6] Moses Charikar, Chandra Chekuri, Tomás Feder, and Rajeev Motwani. Incremental clustering and dynamic information retrieval. In *Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*, STOC '97, pages 626–635, New York, NY, USA, 1997. ACM.

[7] Marina Drosou and Evaggelia Pitoura. Diversity over continuous data. *IEEE Data Eng. Bull.*, 32(4):49–56, 2009.

[8] Marina Drosou and Evaggelia Pitoura. Search result diversification. *SIGMOD Rec.*, 39(1):41–47, September 2010.

[9] Marina Drosou and Evaggelia Pitoura. Dynamic diversification of continuous data. In *Proceedings of the 15th International Conference on Extending Database Technology*, EDBT '12, pages 216–227, New York, NY, USA, 2012. ACM.

[10] Erhan Erkut, Yilmaz Ülküsal, and Oktay Yenicerioğlu. A comparison of p-dispersion heuristics. *Comput. Oper. Res.*, 21(10):1103–1113, December 1994.

[11] Teofilo F. Gonzalez. Clustering to minimize the maximum intercluster distance. *Theor. Comput. Sci.*, 38:293–306, 1985.

[12] Sudipto Guha, Adam Meyerson, Nina Mishra, Rajeev Motwani, and Liadan O'Callaghan. Clustering data streams: Theory and practice. *IEEE Trans. on Knowl. and Data Eng.*, 15(3):515–528, March 2003.

[13] Wenyu Hu and Baili Zhang. Study of sampling techniques and algorithms in data stream environments. In *FSKD*, pages 1028–1034. IEEE, 2012.

[14] Gerold Jäger, Anand Srivastav, and Katja Wolf. Solving generalized maximum dispersion with linear programming. In *Proceedings of the 3rd international conference on Algorithmic Aspects in Information and Management*, AAIM '07, pages 1–10, Berlin, Heidelberg, 2007. Springer-Verlag.

[15] Adam Marcus. Tweeql. https://github.com/marcua/tweeql.

[16] Adam Marcus, Michael S. Bernstein, Osama Badar, David R. Karger, Samuel Madden, and Robert C. Miller. Tweets as data: demonstration of tweeql and twitinfo. In *Proceedings of the 2011 ACM SIGMOD International Conference on Management of data*, SIGMOD '11, pages 1259–1262, New York, NY, USA, 2011. ACM.

[17] Enrico Minack, Wolf Siberski, and Wolfgang Nejdl. Incremental diversification for very large sets: a streaming-based approach. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*, SIGIR '11, pages 585–594, New York, NY, USA, 2011. ACM.

[18] S. S. Ravi, Daniel J. Rosenkrantz, and Giri Kumar Tayi. Facility dispersion problems: Heuristics and special cases (extended abstract). In Frank K. H. A. Dehne, Jrg-Rdiger Sack, and Nicola Santoro, editors, *Algorithms and Data Structures, 2nd Workshop WADS 91, Ottawa, Canada, August 14-16, 1991, Proceedings*, volume 519 of *Lecture Notes in Computer Science*, pages 355–366. Springer, 1991.

[19] Erik Vee, Utkarsh Srivastava, Jayavel Shanmugasundaram, Prashant Bhat, and Sihem Amer-Yahia. Efficient computation of diverse query results. In *ICDE*, pages 228–236, 2008.

[20] Marcos R. Vieira, Humberto Luiz Razente, Maria Camila Nardini Barioni, Marios Hadjieleftheriou, Divesh Srivastava, Caetano Traina Jr., and Vassilis J. Tsotras. On query result diversification. In *ICDE*, pages 1163–1174, 2011.

[21] Michael J. Welch, Junghoo Cho, and Christopher Olston. Search result diversity for informational queries. In *Proceedings of the 20th international conference on World wide web*, WWW '11, pages 237–246, New York, NY, USA, 2011. ACM.

[22] Wikipedia. Reservoir sampling — Wikipedia, the free encyclopedia, 2013. [Online; accessed 3-May-2013].

[23] Cong Yu, Laks Lakshmanan, and Sihem Amer-Yahia. It takes variety to make a world: diversification in recommender systems. In *Proceedings of the 12th International Conference on Extending Database Technology: Advances in Database Technology*, EDBT '09, pages 368–378, New York, NY, USA, 2009. ACM.