# Exploring Alternative Clustering for PIY Source Code Detection

Pa Woua Vang
Computer Science Department
College of Saint Benedict
Saint Joseph, MN 56374
P1VANG@CSBSJU.EDU

James Schnepf
Computer Science Department
Saint John's University
Collegeville, MN 56321
JSCHNEPF@CSBSJU.EDU

## Abstract

In this paper, we compare different clustering algorithms given a specific type of data set. Clustering is a powerful tool because it distributes data into meaningful groups based on the information found in data sets. Given a set of data points, each defined by a set of attributes, we find clusters such that points in one cluster are more similar to one another and less similar to points in other clusters. These groups of clusters are crucial to how data is analyzed. It helps us easily identify and give meaning to certain data according to their traits. Clustering helps handle a data set with more utility thus the study of techniques for finding the most representative cluster model is vital in knowledge extraction. Previous work done by Anthony Ohmann and Professor Imad Rahal propose a scalable system called PIY (Program It Yourself) that can detect source code plagiarism over a large repository of submissions where new submissions are compared to current ones. By using clusters, one can compare a new submission to a subset of the data. Accuracy and time are both important factors for PIY. Therefore, we base efficiency of clustering on accuracy and time. In this paper, we perform an analysis of K-Harmonic Means (KHM) against one of PIY's current clustering algorithms called K-Medoid. Developed by Dr. Bin Zhang, the KHM algorithm is derived from the K-Means and Harmonic Average algorithm. It is known to be more "robust" than the K-Means algorithm. Our goal is to find which algorithm gives us the most favorable results.

# 1. Introduction

Clustering is a powerful method that distributes data into meaningful groups based on the information found in a given data set. Data clustering is often used in statistics, machine learning, and data mining. Clustering is able to identify and group objects that are highly similar to one another. They must also be very different from objects in other groups. The data in each group should share common traits or interesting patterns [2]. Those grouped objects are defined as clusters.

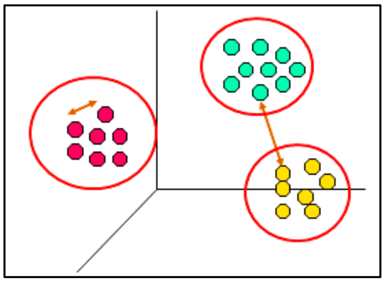

Figure 1: Graphical Representation of 3 Clusters.

The goal of clustering is to determine the intrinsic grouping in a set of unlabeled data. Unlabeled means data objects are not yet identified with a meaning. Benefits to clustering involve simplification, pattern recognition, data construction and unsupervised learning (data mining). This unsupervised step uses all attributes belonging to the objects to make predictions or label data without human involvement. An example of this is a business clustering a collection of customer data to learn about certain shopping habits for additional marketing activities [11].

It is rare to have perfect clusters like the ones shown in Figure 1 [6]. Noise and outliers can disrupt the way data cluster. Noise is the “random component of a measurement error” [10]. Programming errors, hardware failure or human mistakes, in terms of wrong input, are usually the cause to noise. Outliers on the other hand are either data objects with very different characteristics from other objects or one of its attribute values are unusual in comparison to the typical values for that attribute [10]. A visual representation of noise and outliers is given in Figure 2 [6].

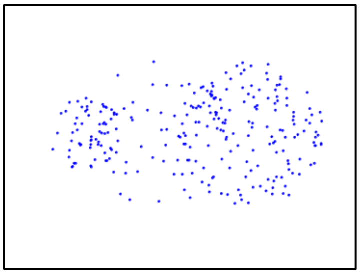

Figure 2: Graphical Representation of 2 Clusters with Noise

There are multiple ways data objects can be grouped due to the dynamics of data. Sometimes the notion of clusters can be indistinct. Therefore, many clustering techniques exist to explore ways data can be grouped, extracted and represented [3].

K-Means is a popular well-known clustering technique. It is a partitional clustering technique that classifies a given set of *n* data objects in *k* clusters where *k* is the number of desired clusters the user must provide [11]. Clusters are formed based on the calculated average. Hence the name K-Means. PIY currently uses a clustering technique fairly close to K-Means called K-Medoid. K-Medoid however uses an actual data object close to a cluster's center called medoid. Through the same partitional attempt, it uses cost to help determine its medoid. In this analysis, we will use and discuss an alternative algorithm to evaluate clustering for PIY. The algorithm we focus on is known as the K-Harmonic Means (KHM) developed by Dr. Bin Zhang. The KHM algorithm is derived from the K-Means and the Harmonic Average function which is another form of mean and will later be further discussed. In general, KHM is known to be more "robust" than the K-Means algorithm [16].

Previous work done by Anthony Ohmann and Professor Imad Rahal propose a scalable system called Program It Yourself, shortened as PIY. It can detect source code plagiarism over a large repository of submissions where new submissions are compared to current ones [10]. The main purpose of PIY is to accurately detect the most likely plagiarized submissions and present it to the user. PIY is built to maintain efficiency over a large repository. Thus by clustering, one can compare a new submission to a subset of previous submissions instead of the entire repository. Using clustering techniques, PIY was able to achieve efficiency in accuracy and time.

Our goal is to find which algorithm will give us the most favorable results in terms of improving PIY's efficiency. Precisely we are interested in evaluating how accurate PIY detects plagiarized forms as well as speed in order to make a supposition on the different clustering techniques we use. We evaluate whether KHM as an algorithm is able to perform efficiently (given our type of data set and goal of our system) in addition to improving on the current clustering in PIY.

After providing background information on our plagiarism system PIY as well as our repository, the basic concept of center-based clustering will be explained followed by PIY's current clustering. Afterwards, KHM will then be defined, an experimental set up will be covered and implementation issues will be discussed. Finally, a concluding analysis of KHM will be made with PIY's current clustering technique K-Medoid.

## 2. Program It Yourself (PIY) Background

Plagiarizing is a crucial misconduct for a student to commit. It is a sign of disrespect to the student's instructor/s and peers. In a setting where there is a large number of students enrolled for a course, it can be hard to catch plagiarized assignments. Thus systems to detect plagiarism has been devised. An example of an online available plagiarism detection software is TurnItIn.com. TurnItIn.com's plagiarism checker uses archived web pages, student submitted papers, published works like textbooks, newspapers, and journals to assist detecting plagiarism in submitted text documents [13]. Nowadays many resources have become easily accessible for almost everyone thanks to tools such as the internet. These consist of more than just articles and books; they also involve source code.

Like TurnItIn.com, in which checks submitted papers for potential plagiarism, there exist source code plagiarism systems. One of the standard systems for detecting source code plagiarism is known as the MOSS. It is fairly easy to change a variable name in a code so that it looks like one's own work. However, the goal for many systems like MOSS is to be able to detect even those sketchy tactics. Therefore Anthony Ohmann and Imad Rahal proposed the PIY plagiarism detection system in order to be more "robust" than systems like MOSS.

## 2.1 Repository

PIY is capable of performing plagiarism detection over a large repository of submissions. Although PIY has the potential of handling different types of source code, submissions PIY is currently structured for Visual Basic source code. This is because the data set we work with involves a case study of an introductory Visual Basic programming course at our institution. Our repository currently consist of approximately 600 student projects submissions. These submissions are a collection of the course's required project assigned to students where they independently choose their topic and apply the skills and techniques they have learned over the course. With the open nature of the project, it is expected for projects to be significantly different from one another [10]. In particular, distinction should be shown in the implementations of student projects.

Visual Basic revolves around a graphical environment. It is a GUI-based programming language where users can easily see and directly program graphical objects. "A Visual Basic project is typically made up of a number of units called forms which resemble classes in object-oriented languages such as Java and C++" [10]. Those forms are known as subroutines. Hence there are three levels to consider when detecting plagiarism. The levels are project-level, form-level and subroutine-level. Project-level focuses on detecting large blocks of code basing plagiarism on the number of similar forms between projects while subroutine-level focuses on smaller blocks of code making the number of comparisons high. In addition, if a submission contains many boilerplate code, results could suffer from projects falsely identified as plagiarized. Form-level plagiarism detection on the other hand compares each form independently where the likeliness of plagiarism is based on the amount of shared code between two forms [10]. Therefore PIY is built to detect form-level plagiarism.

Submissions of Visual Basic projects are processed through numerous steps before comparisons can be made. The submissions are first cleaned (removing indentations and white spaces that could be disguising techniques) and then tokenized in PIY. Tokenization is the process where unique identifiers are assigned to sensitive data yet still representing the essential information of the data needed for processing. For PIY's purposes, tokenizing makes the system more robust against simple disguising tactics in source code. Once a library of tokens is generated, forms of code are read and tokenized into a document. Each of those documents is broken into continuous subsets of *l* length called k-grams and a record of their frequencies are stored to make pairwise comparisons. More information on the tokenization and the k-gram process of PIY submissions can be found in [10].

Since PIY is designed to maintain a large repository of submissions and performs comparisons with each new submission, clustering can greatly reduce the number of pairwise comparisons. Clustering in our case will ultimately create or identify a prototype, a center, to represent its cluster of similar documents. Therefore if there exists a potential plagiarized submission, clustering will make it easier to locate the group of similar projects and run comparisons with that specific group's projects instead of comparing that submission with the whole entire repository.

With the effectiveness of clustering, PIY is able to achieve efficiency. Two current clustering PIY support are the DBSCAN and K-Medoid techniques. DBSCAN is a density based clustering technique where a cluster is determined by how dense the area around it is given an epsilon threshold value and a minimum points value (how many points allowed within the epsilon of a core). Already proven ineffective in [10], DBSCAN's comparison time to K-Medoid was more than double thus we omit it in this paper.

## 3. Center-Based Clustering

There are many applications in cluster analysis that exist to extract knowledge from any data set for utility. There are many clustering methods because the notion of clusters can be uncertain. Figure 3 provides a visual of numerous ways the same set of data can be clustered [6]:

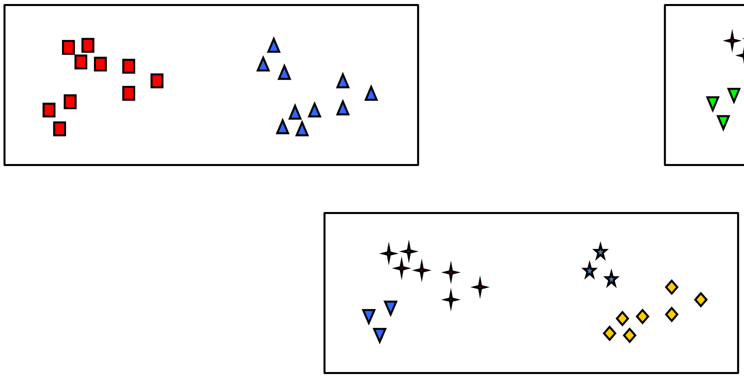

Figure 3: Ambiguous Notion of Clusters

In this paper, we will only focus on one type of clustering. We will focus on center-based clustering methods as opposed to density-based or graph-based clustering. A center-based cluster is a set of objects such that an object in a cluster is closer (more similar) to the "center" of a cluster, than to the center of any other cluster. The center of a cluster is often a centroid, the average of all the points in the cluster, or a medoid, the most “representative” point of a cluster --- i.e. closest to centroid [Tan]. Center-based clustering is also known as prototype-based.

Each clustering has its own objective function. The objective function defines how good a clustering is. The goal of center based clustering is to minimize the objective function [5]. In order to minimize the objective function, since it cannot be done directly, the position of centers are repeatedly refined until it reaches a local optimal. Iteration is used to update the center-based clusterings until it converges. A general algorithm for centered-based clustering defined by [5] is given as follows:

1. Initialize the algorithm with guessed centers C
2. For each data point $x_i$, compute its membership m($c_j|x_i$)
3. For each center $c_j$, recomputed its location from all data point $x_i$, according to their membership and weights:

$$c_j = \frac{\sum_{i=1}^{n} m(c_j|x_i)w(x_i)x_i}{\sum_{i=1}^{n} m(c_j|x_i)w(x_i)}$$

4. Repeat steps 2 and 3 until convergence

“The membership function $m(c_j|x_i)$ defines the proportion of data point $x_i$ that belongs to center $c_j$ with constraints $m(c_j|x_i) \geq 0$ and $\sum_{j=1}^{k} m(c_i|x_i) = 1$” [5]. More precisely, there exists a hard membership and a soft membership. Hard membership means that a specific data object belongs to only one cluster while soft membership means that a data object can belong to different clusters. “A weight function $w(x_i)$ defines how much influence data point $x_i$ has in recomputing the center parameters in the next iteration with constraint $w(x_i) > 0$” [5].

## 4. PIY’s Current Clustering

The number of pairwise comparisons can be greatly reduced with the use of clustering. The clustering technique PIY currently supports is K-Medoid. It is a closely related clustering technique to K-Means. K-Means is well known due to its simple algorithm. The basic algorithm for K-Means is as follows [Pang-Nang]:

```
1. Select K initial centroids
2. Repeat
3.     Form K clusters by assigning each point to its closest centroid
4.     Recompute the centroid of each cluster
5. Until centroids don’t change (converged)
```

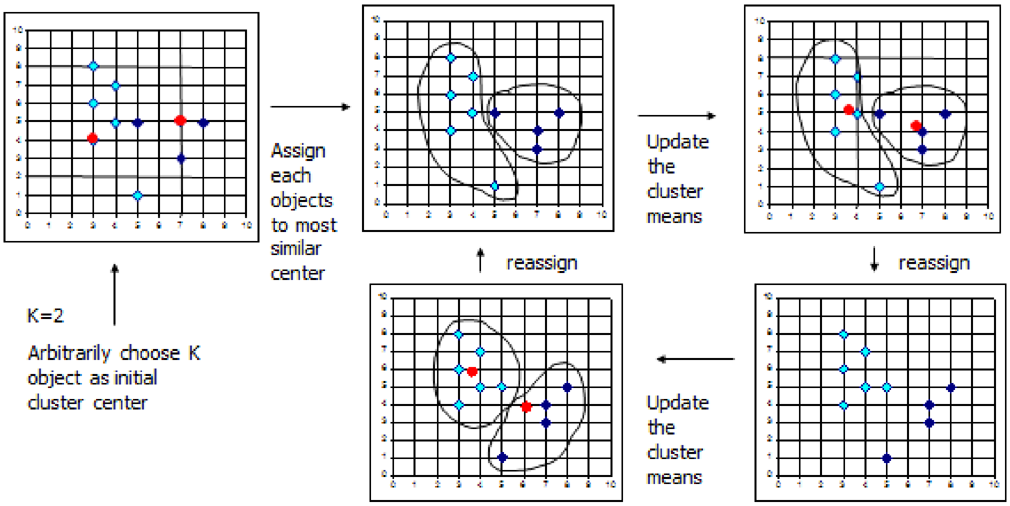

Figure 4: K-Means, k =2

K-Means and K-Medoid are also known as partitional clusterings. Partitional clustering means that the data objects are divided into non-overlapping clusters – no two clusters are sharing the same objects. The specific K-Medoid algorithm used in [10] is known as Partitioning Around Medoid (PAM). A medoid can be defined as an actual representative center, which in our case is an actual project. The PAM algorithm can be defined as follows [7][10]:

1. Randomly choose K initial medoids from all data points
2. Assign all non-medoid to their closest medoid. If the swap can reduce the overall cost (the distance between each non-medoid and the medoid closest to it) of the clustering configuration
3. Repeat steps 2 and 3 until an iteration passes without any medoids changing, i.e, until the algorithm converges

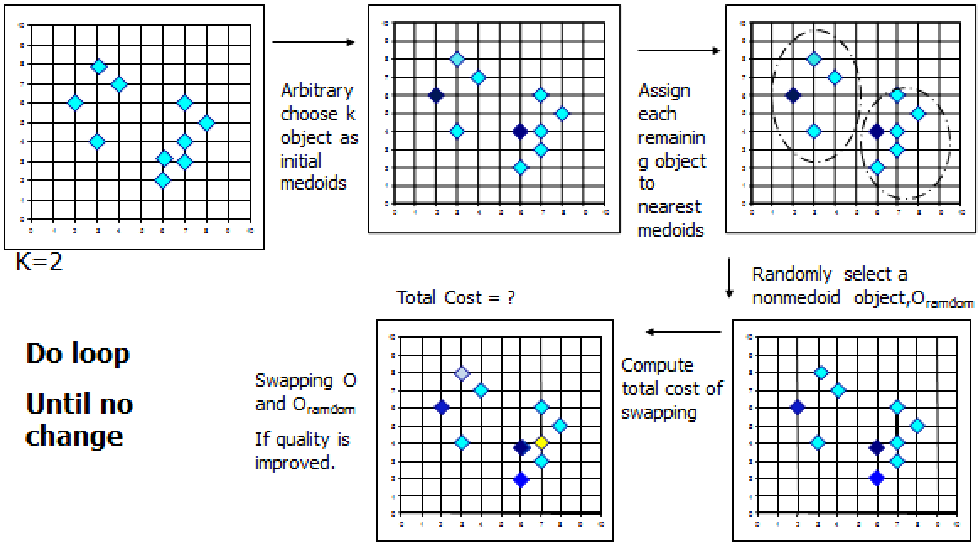

Figure 5: K-Medoid, k =2

Here we note the big difference between K-Means and K-Medoid. K-Medoid takes into account of the cost to swap a medoid. Since a medoid is an actual point per se and not a calculated average, it outdoes K-Mean in the sense that every data object is actually being compared to one another as they are assigned to their closest medoid.

### 4.1 Strengths and Weaknesses of K-Means and K-Medoid

The K-Means clustering technique is popular because it is easy to understand and implement. When it has good initialization, it performs well. This implies that results depend on the initialization. Hence K-Means is sensitive with initialization. The chance of finding the center to every cluster is very low [6]. No specific way to best initialize centroids has been devised. The most popular way is through randomization. K-Means also requires an input value of *k* number of clusters in advance. This is a usual problem in K-Means because there might actually be more than *k* real clusters in a data set. Another problem within K-Means is that clusters are different sizes, more dense than another and non-globular shapes. In addition, noisy data and outliers cause issues to K-Means. A data object of very different values will disrupt the distribution of data [12].

Unlike the K-Means clustering technique, K-Medoid is stronger against noise and outliers. Extreme data objects are less influential to medoids since medoids are actual data objects [12]. The K-Medoid also, however, faces issue with the constraint to provide *k* number of clusters in advance. Lastly, it is relatively not so much efficient compared to the K-Means since initial partition still poses problems.

## 5. K-Harmonic Means Clustering

K-Harmonic Mean is an enhanced version of the K-Means algorithm developed by Dr. Bin Zhang. Due to K-Means well-known problem of sensitivity to initialization (winner takes all partition or greedy algorithm), KHM was developed to break means strong membership of the winner takes all partition that prevents centers from moving out of a local density of data [16]. The Harmonic Mean function boosts the data that are not close to any center by giving them a higher weight in the next iteration. Therefore a special feature present in KHM is its “built-in” dynamic weight function. This means weighting function is automatically adjusted in each iteration. As we have defined earlier in this paper, the weight function is a very important factor in discovering centers such that it helps determine the clusters. Many have proposed different initialization but instead Dr. Zhang attacks the “winner takes all” strategy of K-Means making this algorithm stand out.

### 5.1 Harmonic Mean

To understand KHM, let us first define the concept of the Harmonic Mean function. The Harmonic Mean is best used in situations where extreme outliers exist. Figure 5 depicts a visual of an outlier [9].

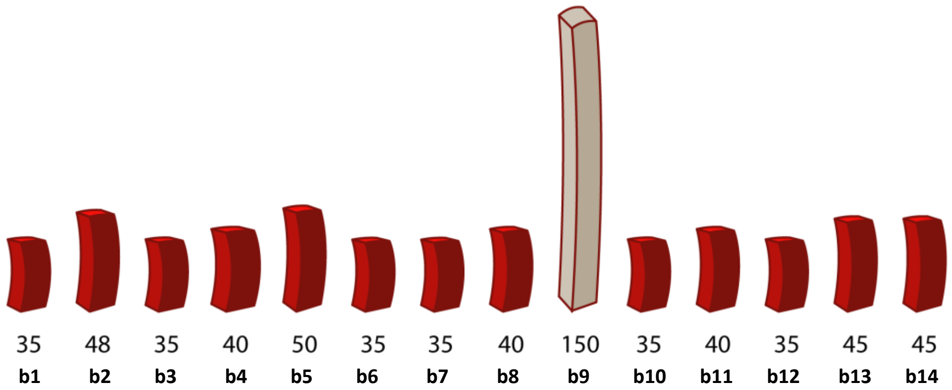

Figure 6: Population

In Figure 6, there is one bar three times larger than the rest of the other bars. This is known as an outlier. Outliers have a bad effect on the Means function. The one bar with a population of 150 will deliver biased results. The Harmonic Mean more preferable in order to best represent the average population [9]. As explained in [9], the arithmetic mean for Figure 5 is approximately 47.7 while the harmonic mean is approximately 41. The harmonic mean provides a more reasonable average of this population, discounting the large outlier by giving less weight to it.

A simple computation of how the Harmonic Mean differs from the Arithmetic Mean is given below:

For Figure 6 bars b6, b7, b8 , b9, b10 the Arithmetic Mean is defined as:

A = (35 + 35 + 40 + 150 + 35)/5
A = 59

While the Harmonic Mean is defined as:

H = 5/(1/35 + 1/35 + 1/40 + 1/150 + 1/35)
H = 42.6

Generally, the Harmonic Mean is can be defined as the total number of data values divided by the sum of the reciprocals of those values (reciprocal of the Arithmetic Mean of reciprocals). We see in the computations above that the outlier b9 with a high value of 150 affects the average quite a bit. The Arithmetic Mean function gave a higher value as opposed to the Harmonic Mean.

## 5.2 KHM algorithm

KHM is a centered-based clustering technique that uses the Harmonic Mean function in its algorithm. Just like K-Means, KHM uses the distance of each data object and centroid to find its clusters. The Harmonic Mean essentially helps overcome the sensitivity of initialization of centers. “The algorithm will naturally shift one or more of these centers to areas where there [are] data point[s] that have no close center” [17] by assigning large weights to those data points. When each data point is close to any one center, a good low score will be computed since the Harmonic Mean is similar to the MIN function as shown in [16].

The KHM function:

$$KHM(X,M) = \sum_{i=l}^{N} HA(||x_i - m_l||^p \mid l = 1, \dots, K) = \sum_{i=l}^{N} \frac{K}{\sum_{l=1}^{K} \frac{1}{||x_i - m_l||^p}} \tag{1}$$

Where $x_i$ is a data point, $m_l$ is a center point and $K$ is the number of clusters. KHM also has the dimensionality parameter value $p$. The higher the dimension of data, the larger $p$ is desired. It is shown that KHM works best with values of $p>2$. The $p$ value helps provide a desired weight function [16]. As we have mentioned earlier in this paper, the weight function is a very important factor in determining how much influence a data object has in the next iteration. With the Harmonic Mean, KHM's special feature of a "built-in" dynamic weighting function is automatically adjusted in each iteration.

The following is a basic algorithm for the KHM clustering [1]:

1. Select an initial partition of the *N* point into *K* clusters by selecting *N* points as initial center randomly. Calculate objective function value according to (1)
2. Obtain membership matrix according to
   $$u_{ij} = \frac{||x_i - c_j||^{-p-2}}{\sum_{j=1}^{K} ||x_i - c_j||^{-p-2}} \tag{2}$$
   An element $u_{ij}$ of this matrix represents the grade of membership of point $x_i$ in cluster $c_j$. Typically $u_{ij}$ within [0,1]
3. Calculate weight of each point according to
   $$w(x_i) = \frac{\sum_{j=1}^{K} ||x_i - c_j||^{-p-2}}{\left(\sum_{j=1}^{K} \left||x_i - c_j|\right|^{-p}\right)^2} \tag{3}$$
4. With membership and weight values of each point calculate new locations of centers according to
   $$c_i = \frac{\sum_{i=1}^{N} \frac{1}{\left(d_{i,k}^{p+2} \sum_{l=1}^{K} \frac{1}{d_{i,k}^{p}}\right)^2} x_i}{\frac{1}{\left(d_{i,k}^{p+2} \sum_{l=1}^{K} \frac{1}{d_{i,k}^{p}}\right)^2}} \tag{4}$$
5. Calculate objective function value with new center values.
6. Repeat step 2-5 predefined number of iterations or until KHM(X, C) does not change significantly
7. Assign point *i* to cluster *j* with biggest $u_{ij}$ value

# 6. Experimental Set-up

Our data set consists of approximately 600 Visual Basic projects. 2935 forms of projects meet the token requirement in [10]. 180 forms has been manually plagiarized to various degrees and used as submissions for testing in [10]'s K-Medoid clustering technique. These 180 manually plagiarized forms will essentially be used to run tests for clustering comparisons. PIY's job is to return a set number (currently set to 200) of pairs of its closest match listing from the most similar to the least. A perfect system would be able to predict and return all 180 plagiarized submissions from their original.

## 6.1 F1 Measure

F1 measure is a common metric to compute a test's accuracy. It uses the precision and recall of the test to compute its score. Precision is the fraction of when PIY labels a project as plagiarized, when was it correct. Recall on the other hand stands for the fraction of successfully retrieved plagiarized forms. Figure 6 shows the formula for the F1 measure, precision and recall. TP stands for True Positives, the number of correctly identified plagiarized projects. FP stands for False Positives, the number of projects mistakenly identified as plagiarized. Lastly, FN stands for False Negative, the number of plagiarized projects that were undetected. Overall, the best F1 score would return a value of 1 while the worst returns 0.

$$P = \frac{\mathrm{TP}}{\mathrm{TP+FP}} \qquad R = \frac{\mathrm{TP}}{\mathrm{TP+FN}} \qquad F1 = \frac{\mathrm{2PR}}{\mathrm{P+R}}$$

Figure 6: Precision *P*, Recall *R*, *F1* Score

## 6.2 Distance/Similarity Measure

In order to compare two submissions, we must find a way to tell how alike and different submissions are from one another. A distance or similarity measure must be chosen to accomplish this step. Distance can be described as the space between two objects or values. Similarity in particular covers how close those two objects are to each other. With a similarity measuring function, a pair more alike will have a higher value. Dissimilarity metric on the other hand, or distance, is going to measure how far apart the two objects are.

There are multiple ways to measure distance. During pairwise comparison tests in [10], two metrics were found to be most efficient given our data: Cosine and Manhattan similarity measures. The Manhattan is originally a distance measure. To use it as a similarity measure, the distance between its two vectors *p* and *q* are normalized to a range 0 and 1 divided by the maximum distance of any vector in the data set. The Manhattan similarity is then retrieved by subtracting the normalized value from one [10]. Since the Cosine and Manhattan metric works fairly well with our data set, it is used to compute the distance as part of K-Medoid's algorithm. We follow this similar fashion regarding the new clustering algorithm KHM that will be analyzed.

## 6.3 Parameters

The authors of [10] found two pairs of suitable parameters, a *l*-value for k-gram processing and a similarity metric, efficient enough for our data set. The Cosine and Manhattan metrics were evaluated as the most effective metrics. F1 scores were measured to depict what was considered the best pair. After the measurement of the F1 scores, various combination of different similarity metrics and *k-gram l*-values, it was reported that Manhattan gave a high F1 score when $l = 4$ and Cosine gave a high F1 score with $l = 13$ (we use $l$ in order to avoid confusion with $k$ clusters). These measurements are runs before K-Medoid's clustering. Therefore, we check KHM's accuracy against K-Medoid.

The best number of clusters for K-Medoid is when $k = 128$. We are interested in solving K-Means and K-Medoid's problem of sensitivity to initialization. Therefore we will specifically look at the different number of $k$ clusterings between K-Medoid and KHM. In addition, we take into account the  dimensionality parameter $p$ in KHM's algorithm. Thus we perform multiple runs of KHM with different values of p>2. [16] has shown that $p$>2 is best used. Running multiple $p$'s allows us to see KHM's performance with our data set. Once all computation is complete, time and accuracy comparisons will be analyzed and determine KHM's best $k$-value.

The best way to run and analyze KHM against K-Medoid involves running each algorithm multiple times. Due to the sensitivity of initializing $k$, we take the best results out of multiple runs. Then we take the average of those results and use it for analysis. Reported in [10], K-Medoid was able to maintain accuracy and decreased in runtime. Thus our goal is to perform better with the possibility of increasing the F1 score with a faster runtime.

## 7. Implementation Issues

PIY saves its submissions to use for future comparisons. When running tests, without the knowledge of how PIY's database work, one can potentially face issues with replicative files which ruins results. Since submissions go through numerous steps of processing before comparisons, each project's generated k-gram processed documents of frequencies are saved into a temporary database directory in the PIY system. When regenerated with the same files for testing, old submissions of the testing files kept within the system. It is best advised, for testing purposes, to empty the temporary database directory in before running every test. Another implementation issue much more arduous is the complexity of the KHM algorithm. KHM's algorithm tends to encounter numerical difficulty with its reciprocals during recursion [15]. In order for successful implementation, calculations must be carefully devised. [15] shows partial steps to overcome the difficult part in KHM's recursion.

## 8. Conclusion

The K-Medoid PAM clustering in our plagiarism detection system PIY is able to give us pleasing results by maintaining accuracy and reducing runtime. Our goal is to find whether the KHM algorithm gives us even more favorable results in terms of improving PIY's efficiency. We are interested in the behavior of accuracy and time for each clustering algorithm. In addition, we want to evaluate whether the KHM clustering technique is able to perform efficiently with our particular data set and improve on the K-Medoid clustering in PIY.

In this paper, we rely mostly upon the previous works of [10] and [16] to evaluate and compare how well the KHM clustering technique scales up to K-Means and PIY's current clustering K-Medoid. We begin by defining the K-Means and K-Medoid clustering techniques well as their strengths and weaknesses. Both K-Means and K-Medoid are sensitive to the initialization of *k* clusters since the centers/medoids of the clusters are randomly chosen. KHM is then presented in hopes to tackle accuracy, reduce the runtime of pairwise comparisons between submissions and resolve the sensitivity to initialization problem. The major difference among K-Means, K-Medoid and KHM is the "built in" dynamic weight function that exists in KHM. It has the ability to affect how much a certain data object should (weigh) participate in recalculating the new centroid based on how close it is to a center. This "built in" makes calculating the centroid in the next iteration more coherent. Therefore it aids in avoiding numerous clusters in one densely packed area of data.

We plan to extend our approach by running experimental tests between the KHM and K-Medoid clustering algorithms. Analysis and evaluations will be performed to illustrate which clustering will truly work with our system PIY and its current data set.

# References

[1] Alper Ü and Zülal G (2009), Applying K-harmonic means clustering to the part-machine classification problem, Expert Systems with Applications, Volume 36, Issue 2, Part 1, Pages 1179-1194

[2] Bankapalli J and Venu Babu R and Anjali Devi S (2011). A Combining K-Harmonic mean and Hierarchical algorithms for robust and efficient data clustering with cohesion self-merging, International Journal on Computer Science and Engineering (IJCSE), vol. 3, no. 6, pp. 2544-53

[3] Estivill-Castro V (2002). Why so many clustering algorithms: a position paper. SIGKDD Explor. Newsl. 4, 1, pp. 65-75

[4] Finkelstein, M. O (2009). Basic Concepts of Probability and Statistics in the Law. New York: Springer, pp. 25-27

[5] Hamerly G and Elkan C (2002). Alternatives to the k-means algorithm that find better clusterings. In Proceedings of the eleventh international conference on Information and knowledge management (CIKM '02). ACM, New York, NY, pp. 600-607

[6] Imad R.Clustering [Power Point Slides]. Retrieved from St. John's University CS332 course material.

[7] Kaufman L and Rousseeuw PJ (1990). Finding Groups in data: an introduction to cluster analysis. John Wiley & Sons, New York, NY

[8] Maimon, O. Z., & Rokach, L. (2005). *Data mining and knowledge discovery handbook*. New York: Springer, pp. 322-52

[9] Matuszak, A. (2010). Differences between Arithmetic, Geometric, and Harmonic Means. Retrieved March 21, 2014, from

http://economistatlarge.com/finance/applied-finance/differences-arithmetic-geometric-harmonic-means

[10] Ohmann A (2013). Efficient Clustering-based Plagiarism Detection using IPPDC, Honors Theses, Paper 14, Retrieved from http://digitalcommons.csbsju.edu/honors_theses/14

[11] Pang-Ning T, Steinback M, and Kumar V (2006). Introduction to Data Mining, Addison-Wesley, Boston, MA, pp. 487-555

[12] Singh, S. S., & Chauhan, N. C. (2011). K-means v/s K-medoids: A Comparative Study. In National Conference on Recent Trends in Engineering & Technology

[13] TurnItIn (2014). Retrieved March 22, 2014 from http: www.turnitin.com

[14] Zhang, B (2003). Comparison of the performance of center-based clustering algorithms. In *Advances in Knowledge Discovery and Data Mining*, pp. 63-74

[15] Zhang B, Hsu M, Dayal U (1999). K-Harmonic Means – a data clustering algorithm, Technical Report HPL-1999-124, Hewlett-Packard Laboratories

[16] Zhang B (2001). Generalized K-Harmonic Means - Dynamic Weighting of Data in Unsupervised Learning, the First SIAM International Conference on Data Mining (SDM'2001), Chicago, USA, April 5-7

[17] Zülal G and Alper Ü (2008), K-Harmonic means data clustering with tabu-search method, Applied Mathematical Modelling, Volume 32, Issue 6, Pages 1115-1125