Non-Exhaustive, Overlapping k-medoids for Document Clustering

Eric Kerstens
Bentley University
kersten_eric@bentley.edu

Abstract

Manual document categorization is time consuming, expensive, and difficult to manage for large collections. Unsupervised clustering algorithms perform well when documents belong to only one group. However, individual documents may be outliers or span multiple topics. This paper proposes a new clustering algorithm called non-exhaustive overlapping k-medoids inspired by k-medoids and non-exhaustive overlapping k-means. The proposed algorithm partitions a set of objects into k clusters based on pairwise similarity. Each object is assigned to zero, one, or many groups to emulate manual results. The algorithm uses dissimilarity instead of distance measures and applies to text and other abstract data. Neo-k-medoids is tested against manually tagged movie descriptions and Wikipedia comments. Initial results are primarily poor but show promise. Future research is described to improve the proposed algorithm and explore alternate evaluation measures.

1. Introduction

Text document collections contain large quantities of information that can be difficult to evaluate. These collections appear in contexts such as library catalogues, online articles, and open-ended survey responses. Documents must undergo expensive categorization before people can access underlying information. Unlike categorical and numeric data, statistical and descriptive measures do not apply to free-form text. The data must be labeled with codes, which are brief terms that capture the underlying meaning of language-based data. For example, "Security" describes the following text from *The Coding Manual for Qualitative Researchers*:

> I notice that the grand majority of homes have chain link fences in front of them. There are many dogs (mostly German shepherds) with signs on fences that say "Beware of the Dog" [1].

Codes facilitate categorization, theory building, and pattern-detection [1].

Codes are produced through a two-step process that grows in complexity, time, and difficulty proportional to the document count. Researchers first read through text to determine common themes. These themes become codes in the codebook. Coders then reread every document and assign relevant codes. It becomes harder to track the breadth of ideas as the codelist expands. This can quickly become too time consuming and expensive.

Text is rich with information, but analysis is hindered by the need for time consuming coding. Can machine learning aid researchers and increase the value of these questions by either fully or partially automating response coding? Automation would make open-ended response analysis less daunting and could increase adoption.

1.1. Automated Open-Ended Response Coding

Past research identified the value of automated survey coding and proposes solutions. These solutions attempt to augment coding in various ways. A primary focus is to alleviate the coding burden, either through semi or full automation of response grouping. Other approaches attempt to enhance analysis by accounting for respondent demographics and other sources of information.

Research shows that classification models are viable semi-automated tools that can guide code assignment. Card and Smith use logistic regression models and recurrent neural networks (rNNs) to classify pre-coded open-ended responses from the American National Election Studies (ANES) data set. Card and Smith found that logistic regression models generated consistently useful predictions but rNNs performed poorly [2]. These are encouraging but the process still requires a codebook and coding a large subset of data. Codes are only automatically assigned to remaining data.

Roberts et al. propose an alternative semi-automated approach to coding through unsupervised structural topic models (STMs). STMs treat text as a mix of topics; words are associated with topics to varying degrees. The

URI: https://hdl.handle.net/10125/63836
978-0-9981331-3-3

strength of STMs is that topic-word probabilities vary based on selected covariates. For example, two identical survey responses, one written by a twenty year old male, the other by an eighty year old female, may belong to different topics due to the context of the writer. STMs assign responses to the most probable topics. STMs pull topics from the data and assign related topics via unsupervised learning. Researchers are not required to code documents, but are still heavily involved; they must select the variables to use as covariates and select the final model.

STMs perform well relative to manual codes. Topics are more specific, but small categories are lost. In some cases the model provides descriptive topic labels, but in others labels are unclear. STMs perform well except for two primary limitations: topics may be difficult to interpret and a high degree of user input is required.

Efforts to automate coding achieved a measure of success but are limited in the degree to which they relieve the burden of the process. Attempts to improve automated coding should:

1. Produce concept groups similar to those produced by a team of coders.

2. Produce meaningful labels researchers can use to understand groups. An uninterpretable model does not provide insight into core ideas.

3. Identify documents that address multiple topics and assign multiple codes accordingly. Documents often span multiple separate themes and an automated coding algorithm should catch these.

4. Some documents do not contain noteworthy information; they may be unintelligible or irrelevant [3]. In statistics terminology, these documents are outliers. They should be identified and removed to avoid affecting other analysis.

5. Demand minimal researcher intervention. The largest benefit of automation is reduction in labor. This benefit is nullified if the researcher spends equal time optimizing the model.

This report explores whether document clustering can automate coding to the above specification. Section 2 reviews existing document clustering research and limitations. Section 3 introduces neo-k-medoids, a new document clustering algorithm geared specifically toward the unique characteristics of the coding problem. Section 4 outlines the application of neo-k-medoids to IMDB movie genres and toxic Wikipedia comments; the results are discussed in Section 5. Section 6 summarizes key material and outlines avenues of future research.

2. Document Clustering

At its core, coding groups sets of similar text that share important properties. These properties are the key concepts researchers choose to include in their codebook. The end result is similar to clustering, a form of unsupervised machine learning that divides data into meaningful groups. Similar objects are grouped together and dissimilar objects are separated [4, 5, 6]. The technique is frequently used for document organization. Document clustering is explored in depth as a potential automated coding solution due to the similarity of end results produced by these techniques and manual coding.

The architecture of document clustering has three primary components: text parsing, similarity estimation, and mining [7]. Parsing converts documents into term-frequency vectors between which similarity measures exist. Similarity estimation measures the dissimilarity between documents pairs. Mining is the final step that partitions data. Mining uses the similarity measures to partition data into a set of k clusters $C = \{c_1, \cdots, c_k\}$. This section explores approaches to these three steps to illuminate existing techniques and to explore how they can extend to document coding.

2.1. Text Parsing

Text parsing is the first step of clustering. Documents are abstract and difficult to work with directly; structure is imposed onto the inherently unstructured text to make it easier to extract information. It is typical to represent text as an unordered collection of words called a *bag of words*. A bag is a set that permits duplicate entries. Reduction to a bag of words strips text structure but enables information extraction from a collection of smaller, distinctly meaningful objects [6]. This bag is transformed into the vector space model in which a matrix relates documents to their contents.

Let $D = \{d_1, \cdots, d_N\}$ be the set of unique documents (each represented as a bag of words) under consideration. This collection D of documents is called a corpus. Let $T = \{t_1, \cdots, t_m\}$ be the set of all unique terms in D. Let $\text{tf}(d, t)$ represent the frequency with which term $t \in T$ repeats in document $d \in D$. Each document is represented as an m dimensional vector $\vec{t_d} = \{\text{tf}(d, t_1), \cdots, \text{tf}(d, t_m)\}$ [5]. The set of documents D can be written as a document-term matrix X. Let $X = [x_{ij}]_{n \times m}$ where entry $x_{ij} = \text{tf}(d_i, t_j)$ [8].

Preprocessing converts documents into term frequency vectors. Representing a document as a bag of words requires tokenization, a process that identifies individual terms. Tokenization needs to account for language's irragularities. For example, it

Table 1. Common notation

Notation		Meaning
D	$= \{d_1, d_2, \cdots, d_N\}$	Corpus containing N documents
T	$= \{t_1, t_2, \cdots, t_m\}$	Set of m unique terms in D
$\text{tf}(d, t)$		Term frequency of term t in document d
C	$= \{C_1, C_2, \cdots, C_k\}$	Set of k clusters produced by document clustering
$\lvert\cdot\rvert$		Set cardinality; indicates the number of set elements
$\lVert\cdot\rVert$		L^2 vector norm; indicates vector length
$d(\cdot, \cdot)$		Distance between two objects
$r(\cdot, \cdot)$		Dissimilarity between two objects; may not be a metric

must differentiate between *Wash.*, an abbreviation of state of Washington, and the verb *wash* [6]. There is no generally accepted single model for tokenization deemed to understand the complexity of language.

The vector space model removes document structure, ignoring the important role context plays in the meaning of a word or phrase [6]. Part of this context is retained when collocations are represented in the bag of words. A collocation is two or more words whose meaning extends beyond the composed meaning of its parts [6]. For example, consider the following sentences:

1. Bundling the weeks together was the last straw for Tom Crow.

2. The last weeks Tom was bundling together straw for the crow.

These sentences reduce to the same bag of words despite different meanings unless the idiom "the last straw" and proper noun "Tom Crow" are treated as individual terms.

One way to identify collocations is with N-grams which are series of n words. For example "Tom Crow" is a bi-gram. N-grams provide context to the individual words based on the Markovian assumption that the prior local context of a word provides sufficient evidence to suggest its meaning [6]. Only frequently occurring n-grams are included because additional terms expand the feature space and model complexity.

Specific, informative terms are more useful for identifying core themes. Stop words such as *a*, *from*, and *the* contain minimal information and are often removed [5]. Inverse document frequency (IDF) weights terms based on specificity [7]. IDF is the ratio between the number of documents over the number of documents in which a term appears [7, 6, 9, 5], as follows:

$$\text{idf}(t) = \frac{N}{|\{d : d \in D, t \in d\}|} \tag{1}$$

Term frequency and inverse document frequency are combined with $\text{tfidf}(d, t) = \text{tf}(d, t)\text{idf}(t)$.

There are many available processing steps that convert documents into term-frequency vectors. Alternate steps, such as stemming and lemmatization exist as well. See [6, 10] for more information. Some or all of the above techniques should be implemented before calculating document similarity.

2.2. Similarity Measures

It becomes possible to measure dissimilarity between document pairs after text is converted into term-frequency vectors. These similarity measures provide clustering algorithms a way to determine whether two documents belong in the same or separate partitions. Various measures exist to calculate the distance or dissimilarity between documents. These can be viewed in the context of metrics. Let x and y be two objects in a set. A function $d(x, y)$ is a metric and measures the distance between x and y if and only if it satisfies:

1. $d(x, y) \geq 0$, i.e. the distance between x and y cannot be negative.

2. $d(x, y) = 0 \iff x = y$, i.e. the distance between x and y equals zero if and only if the two objects are identical.

3. $d(x, y) = d(y, x)$, i.e. the distance must be symmetric, such that the distance from x to y must be the same as the distance from y to x.

4. For any third object z in the set, $d(x, z) \leq d(x, y) + d(y, z)$, i.e. the measure must satisfy the triangle inequality.

Some document similarity measures meet all of the above requirements and are considered distance measures. Others, referred to as dissimilarity measures, meet only some criteria [5].

The Euclidian distance between two term frequency vectors, given by:

$$D_E(\vec{t_1}, \vec{t_2}) = \sqrt{\sum_{i=1}^{m} (\vec{t_{1i}} - \vec{t_{2i}})^2} \tag{2}$$

is one distance measure that is a true metric and satisfies the above requirements [6].

Cosine similarity is a dissimilarity measure used to compare documents. It measures the angle between two term frequency vectors [6], and is calculated with:

$$SIM_C(\vec{t_a}, \vec{t_b}) = \frac{\vec{t_a} \cdot \vec{t_b}}{\|\vec{t_a}\| \times \|\vec{t_b}\|} \tag{3}$$

$\vec{t_a}$ and $\vec{t_b}$ are both positive, so SIM_c is non-negative and bounded by $[0, 1]$. Cosine similarity violates the second criterion because if $\vec{t_a} = \vec{t_b}, SIM_c = 1 \neq 0$. Cosine similarity can be rewritten to measure dissimilarity by $D_C = 1 - SIM_C$. This is still not a true metric because two documents d and d' differing in length but equal in term proportions will return $D_C(d, d') = 0$ even though the objects are not identical [5]. Many other measures exist; interested readers should explore the Jaccard coefficient, average Kullback-Leibler divergence, and Dice coefficient, among others [11, 5, 6, 12].

Vector space distance and dissimilarity measures ignore term context, synonyms, and polysemous terms. A polysemous term is one with multiple, disjoint meanings [7]. Alternatives enrich document similarity measures with semantic relationships. Some measures disambiguate core concepts with WordNet; others leverage a large corpus to provide insight into term co-occurrence. See [13, 7, 9] for more detail. However, these measures are rarely used in document clustering; the techniques are more computationally expensive and many focus only on isolated terms. Although semantic similarity measures acknowledge a wider context, the many disadvantages have prevented adoption.

2.3. Clustering Algorithms

Clustering algorithms form the core of document clustering and partition data into meaningful groups. These algorithms are classified as hard, soft, or disjunctive. Hard algorithms assign each document to one and only one cluster. Soft techniques also assign each document to only one cluster but convey a degree of uncertainty about which cluster is correct. Disjunctive clustering models explicitly assign some documents to multiple clusters [6]. There are many popular hard techniques and few published disjunctive algorithms.

It is typical to characterize clusters by their centers, with objects assigned to the cluster with the most similar center. Centroids and medoids form most centers [14]. A centroid is the mean of cluster members [15] and typically does not represent an $x \in X$ [14]. Medoids are specific cluster members that are most representative of other cluster members [6].

k-Means k-means is a popular hard algorithm. For a set of observations $X = \{x_1, \cdots, x_n\}$, k-means generates k clusters $C = \{C_1, \cdots, C_k\}$ which satisfy the following conditions [16]:

$$C_i \neq \varnothing, \quad \forall C_i \in C \tag{4a}$$
$$C_i \cap C_j = \varnothing, \quad \forall C_i, C_j \in C \tag{4b}$$
$$\bigcup_{C_i \in C} C_i = X \tag{4c}$$

and minimize the objective function [17, 18]:

$$J(C, X) = \sum_{i=1}^{n} \sum_{j=1}^{k} z_{ij} d(x_i, \mu_j) \tag{5}$$

where μ_j is the centroid of cluster C_j, where $d(x_i, \mu_k)$ is the Euclidean distance between x_i and μ_j, and where z_{ij} is the indicator variable:

$$z_{ij} = \begin{cases} 1 & if x_i \in C_j \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

In other words, k-means partitions a dataset into k non-empty clusters that minimize the within cluster sum of squared distances. Each data point belongs to only one cluster, and no cluster is empty.

k-means depends on an average. This makes sense for numeric data, but not text. An average term-frequency vector μ_j equal to the average of document vectors in cluster C_i can be used. However, this average has no implicit meaning. It is not possible to cluster with a semantic dissimilarity measure not based in the vector space model.

k-Medoids k-medoids is similar to k-means except cluster centers are medoids. Medoids are selected so that they reduce the average dissimilarity of all objects within the cluster; $m_j = \underset{x_i \in C_j}{\arg\min} \sum_{x_l \in C_j} r(x_l, x_i)$. This choice minimizes the objective function:

$$J(C, X) = \sum_{i=1}^{n} \sum_{j=1}^{k} z_{ij} r(x_i, m_j) \tag{7}$$

where m_j is the medoid of cluster C_j, where $r(x_i, m_j)$ is the dissimilarity between x_i and m_j, and where z_{ij} is defined as before [19].

Note that the objective function for k-medoids defines $r(x_i, x_j)$ as a measure of *dissimilarity*, rather than of distance. $r(x_i, m_j)$ does not have to be a metric. The use of representative objects as cluster centers generalizes k-medoids to non-interval scaled data [19].

Hard clustering is problematic for text because a document can span multiple topics. Soft and disjunctive clustering provide alternatives that allow objects to belong to multiple clusters. Soft methods weight the relationship between each object-cluster pair. Disjunctive clusters explicitly assign objects to multiple clusters.

NMF Non-negative matrix factorization (NMF) is a soft clustering algorithm based on decomposing the document-term matrix. Wei, Liu, and Gong propose NMF for document clustering [8]. NMF approximates a non-negative $m \times n$ matrix X by the product of non-negative matrices $U_{m\times k}$ and ${V^T}_{k\times n}$. U and V^T are the matrices which minimize the objective function:

$$J = \frac{1}{2}\|X - UV^T\| \tag{8}$$

where $\|\cdot\|$ is the squared sum of matrix elements. This optimization is rewritten as a constrained optimization problem and solved via the Lagrange multiplier method.

NMF is applicable to document-term matrices because document frequencies are always non-negative. In document clustering, m is the number of terms, n is the number of documents, and k is the number of clusters.

An entry u_{ij} of U represents the degree to which term $t_i \in T$ is associated with cluster C_j. Each entry v_{ij} of V^T represents the degree to which document $d_i \in D$ is associated with cluster C_j. The degree of association between a document and cluster measures the strength of the relationship between a document and cluster. Documents are assigned to the cluster with which they have the highest association. Documents could be assigned to multiple clusters showing high membership, but no techniques are explored. Cluster labels are based on the terms most strongly related with each cluster.

NMF performs well on manually coded text data. Xu, Liu, and Gong used NMF to divide a collection of news articles into 56 clusters. Generated clusters matched those generated manually with over 65% accuracy. One concern is that NMF generates clusters using all documents without accounting for outliers. Unrepresentative documents may bias resulting clusters.

Although soft clustering can assign points to multiple clusters, each point is typically assigned to one and only one cluster. There is a degree of uncertainty of how strongly a point is connected with its assigned cluster [6]; this is valuable to any researchers who want to know which responses are the most representative of a group. The degrees of uncertainty can also be used to determine outliers or to assign documents to multiple clusters, but in practice it can be difficult to determine the appropriate cutoff points and to control results [15].

Neo-k-means Disjunctive clustering most closely emulates the human coding process by explicitly assigning objects to multiple clusters. The literature makes minimal mention of disjunctive clustering. Non-exhaustive overlapping k-means (neo-k-means) is one such algorithm.

Whang, Hou, Gleich, and Dhillon propose neo-k-means, an overlapping clustering technique that considers outliers [15]. Neo-k-means introduces explicit model parameters that control cluster overlap and the number of expected outliers. Let α control the total number of cluster assignments and require $0 \leq \alpha \leq (k-1)$. Then,

$$\sum_{C_j \in C} \sum_{x_i \in X} z_{ij} = (1+\alpha)n \tag{9}$$

ensures a total of $(1+\alpha)n$ assignments. $\alpha \ll (k-1)$ prevents assigning every point to every cluster [15].

Now let,

$$\mathbb{I}(x_i) = \begin{cases} 1 & \text{if } \sum_{C_j} z_{ij} = 0 \\ 0 & \text{otherwise} \end{cases} \tag{10}$$

indicate whether or not a given object x_i belongs to any cluster, such that $\mathbb{I}(x_i) = 1$ for outliers. The parameter β controls the proportion of outliers in the data, such that there are at most βn outliers. In other words,

$$\sum_{x_i \in X} \mathbb{I}(x_i) \leq \beta n \tag{11}$$

is required. There should only be a small proportion of outliers, suggesting $0 \leq \beta \ll 1$ [15].

Selecting correct values for α and β is critical to cluster performance. Whang, Hou, Gleich, and Dhillon propose that the number of outliers be determined with the results of k-means. The mean μ and standard deviation σ of distances between objects and corresponding centroids are calculated. Any points with a distance greater than $\mu + 3\sigma$ are considered outliers. The proportion of these outliers estimates β. Determining α is more difficult, since it is unclear from the data how

much overlap to expect. The choice may be based on researcher expectations. A suggested value is $\alpha = \sqrt{k}-1$ for a large number of clusters [15].

Neo-k-means was evaluated against a variety of manually coded numeric data sets. When compared against other overlapping algorithms, Neo-k-means had the highest F_1 score in almost all cases [15].

Neo-k-means satisfies many of the requirements of an automated coding algorithm. The algorithm is robust and outlier aware; cluster centers are not biased towards aberrant data. Additionally, the disjunctive algorithm assigns objects to multiple clusters to reflect the idea that a single object can cover multiple concepts. However, in its current implementation neo-k-means works only for numeric data and does not extend to documents.

3. Neo-k-Medoids

Raw text encodes valuable information in many contexts. However, required coding is time consuming and draws researchers and their budgets away from other important work. Attempts to automate coding do not fully emulate the existing process. Human coders produce meaningful labels, potentially multiple per response, and are able to identify outliers. No automated solution has been proposed for the coding problem that meets these criteria.

The neo-k-means algorithm proposed by Whang, Hou, Gleich, and Dhillon [15] identifies outliers and produces disjunctive clusters. These attributes are desirable in the case of document clustering but the algorithm is only applicable to numeric data. We propose non-exhaustive, overlapping k-medoids (neo-k-medoids) a new algorithm based on the principals of neo-k-means but with medoids instead of centroids.

3.1. Neo-k-Medoids Objective Function

Neo-k-medoids aims to partition a set of n objects $X = \{x_1, x_2, \cdots, x_n\}$ into k cohesive clusters $C = \{C_1, C_2, \cdots, C_k\}$ that minimize the objective function:

$$J(C, X) = \sum_{j=1}^{k} \sum_{x_i \in C_j} r(x_i, m_j) \tag{12}$$

where m_j is the medoid of cluster C_j and where $r(x_i, m_j)$ is the dissimilarity between x_i and m_j. Two parameters α and β are introduced that differentiate this algorithm from k-medoids.

β enforces non-exhaustiveness and represents the proportion of outliers. There should only be a small proportion of outliers and so $0 \leq \beta \ll 1$ is suggested. Let $\mathbb{I}(x_i) \in \{0, 1\}$ indicate 1 if x_i is an outlier and 0 if x_i belongs to at least one cluster. Then $\sum_{x_i \in X} \mathbb{I}(x_i) \leq \beta n$ ensures that there are no more than βn outliers.

The α parameter enables cluster overlap. Hard clustering algorithms assign each object to only one cluster with n assignments. In neo-k-medoids there are $(1+\alpha)n$ assignments. In essence, α controls the average number of assignments per object. If $\alpha > 0$ some objects are assigned to multiple clusters.

Note that if $\alpha = \beta = 0$, neo-k-medoids becomes k-medoids. Each object is assigned to exactly one cluster with no overlap or outliers. Under these circumstances other algorithms are more efficient and preferred.

Neo-k-means requires $0 \leq \alpha \leq (k-1)$. If $\alpha = 0$ there are n total assignments even if $\beta > 0$. Either every object is assigned to a cluster (no outliers) or some objects have multiple assignments. It should be possible to produce non-exhaustive, non-overlapping clusters. The restriction is loosened in neo-k-medoids to $-\beta \leq \alpha \leq (k-1)$. If $\alpha = -\beta$ there are $(1-\beta)n$ non-overlapping assignments with βn outliers. $\alpha \ll (k-1)$ is recommended to prevent assigning every object to every cluster.

The objective function becomes:

$$\begin{aligned} J(C, X) &= \sum_{j=1}^{k} \sum_{x_i \in C_j} r(x_i, m_j) \\ \text{such that} \sum_{j=1}^{k} \sum_{x_i \in C_j} 1 &= (1+\alpha)n \\ \text{and} \sum_{x_i \in X} \mathbb{I}(x_i) &\leq \beta n \end{aligned} \tag{13}$$

with the additional constraints.

3.2. The Neo-k-Medoids Algorithm

It would be impractical to search through all possible combinations of medoids, of which there are $\binom{n}{k}$ possibilities, to find the optimal set corresponding with the global minimum of the objective function. Neo-k-medoids instead searches for local minima. Precision is lost at the benefit of practical computation speeds.

The algorithm is comprised of three components: medoid selection, cluster assignment, and medoid swap.

1. **Medoid selection** A set of initial medoids is needed before clustering can begin. k random elements are drawn from X to serve the role of these initial medoids.

2. **Cluster Assignment**

Algorithm 1: The neo-k-medoids algorithm

Parameter : $X :=$ set of n objects $\{x_1, x_2, \cdots, x_n\}$
Parameter : $D := [r(x_i, x_j)]_{n \times n}$ dissimilarity matrix
Parameter : $k :=$ number of clusters. $k \leq n$ required
Parameter : $\beta :=$ outlier parameter. $0 \leq \beta \leq 1$ required
Parameter : $\alpha :=$ overlap parameter. $-\beta \leq \alpha \leq k - 1$ required
Parameter : $t_{\max} :=$ maximum iterations before stopping
Parameter : tolerance := improvement cutoff to terminate search

Output: Set of clusters $\{C_1, C_2, \cdots, C_k\}$

initialize random cluster medoids M = $\{m_1, m_2, \cdots, m_k\}$ for $m_j \in X$
initialize $t \leftarrow 0$
while *not converged* **and** $t < t_{max}$ **do**
  initialize $\tau \leftarrow \varnothing, S \leftarrow \varnothing, a \leftarrow 0, \bar{C}_j \leftarrow \varnothing, \hat{C}_j \leftarrow \varnothing, \forall j$
  `// assign objects to clusters`
  while $p < (n + \alpha n)$ **do**
    if $p < (n - \beta n)$ **then**
      assign x_{i*} to $\bar{C_{j*}}$ such that $(i*, j*) \leftarrow \arg\min_{i,j}(r(x_i, m_j))$ where $\{(i,j)\} \notin \tau, i \notin S$
      $S \leftarrow S \cup \{i*\}$
    else
      assign x_{i*} to $\hat{C_{j*}}$ such that $(i*, j*) \leftarrow \arg\min_{i,j}(r(x_i, m_j))$ where $\{(i,j)\} \notin \tau$
    end
    $\tau \leftarrow \tau \cup \{(i*, j*)\}$
    $a \leftarrow a + 1$
  end
  $\forall j$ update cluster $C_j \leftarrow \bar{C}_j \cup \hat{C}_j$
  `// swap the medoid and object with the greatest intra-cluster dissimilarity improvement`
  initialize $\Theta \leftarrow \{\sum_{x_l \in C_j} r(x_l, m_j)\}_{j=1}^{k}$
  initialize $\Xi \leftarrow [\xi_{ij}]_{n \times k}$ where $\xi_{ij} \leftarrow \sum_{x_l \in C_j} r(x_i, x_l)$
  if $\max_{i,j}(\theta_j - \xi_{ij}) < 0$ **then**
    break `// no medoid swap improves the objective function`
  end
  assign $m_{j*} \leftarrow x_{i*}$ such that $(i*, j*) \leftarrow \arg\max_{i,j}(\theta_j - \xi_{ij})$ where $x_i \notin M$
  $t \leftarrow t + 1$
end

- For each x_i, determine the nearest medoid m_j. Assign $n - \beta n$ documents with the closest medoid to corresponding clusters to ensure at most βn outliers.
- Until there are $n + \alpha n$ total assignments, assign object x_i to cluster C_j satisfying $\arg\min_{i,j}(r(x_i, m_j))$ and $x_i \notin C_j$.

3. **Medoid Swap** Search for the non-medoid that would result in the greatest improvement to the objective function if it were a medoid. Set this object as the medoid of the given cluster. For each cluster, the sum of dissimilarities between medoid and cluster members is $\theta_j = \sum_{x_l \in C_j} r(x_l, m_j)$. For all $x_i \in X$ that are not medoids, let $\xi_{ij} = \sum_{x_l \in C_j} r(x_l, x_i)$ represent the sum of dissimilarities within cluster C_j if x_i were the medoid. Object x_i becomes the medoid of cluster C_j for $(i, j) = \arg\max_{i,j}(\theta_j - \xi_{ij})$. This choice of x_i and C_j results in the greatest overall improvement to the objective function.

The algorithm repeats steps two and three until

converging at a local minimum. Clusters form a local minimum if no medoid swap exists that improves the objective function. If a swap is made the process returns to the assignment stage. See Algorithm 1 for a formalization of this procedure. A Python implementation is demonstrated in [20].

To guarantee convergence to a local minimum, the neo-k-medoids algorithm must monotonically decrease the objective function.

Theorem 1. *The neo-k-medoids algorithm monotonically decreases the objective function $J(C, X)$.*

Proof. Let $J^{(t)} = J(C^{(t)}, X)$, where $C^{(t)}$ represents the set of clusters at the start of the tth iteration. Let $\{m_{j*}\}_{j=1}^{k}$ represent the new medoids after the swap stage. Then,

$$J^{(t)} = \sum_{j=1}^{k} \sum_{x_i \in C_j^{(t)}} r(x_i, m_j) \tag{14}$$

$$\geq \sum_{j=1}^{k} \sum_{x_i \in C_j^{(t)}} r(x_i, m_j*) \tag{15}$$

based on the selection of m_{j*}

$$\geq \sum_{j=1}^{k} \sum_{x_i \in C_j^{(t+1)}} r(x_i, m_j) \tag{16}$$

since cluster assignments change $\Leftrightarrow$ they decrease $r(\cdot)$

$$= J^{(t+1)} \tag{17}$$

$\therefore J(C, X)$ decreases monotonically. □

The monotonicity of neo-k-medoids guarantees convergence at a local minimum relative to the initial set of medoids. In a large sample space many local minima will exist. Optimal results follow from the selection of the best final result of multiple initializations.

4. Methodology

Neo-k-medoids is tested against manually labeled text datasets to evaluate effectiveness. The first dataset describes movies listed on IMDB [21]. The dataset includes movie titles, plots, and genres. Genres are used as manual classes. The data is clustered twice, once on movie titles and again with plots. The second dataset is comprised of comments from Wikipedia's talk page edits [22]. These comments have been flagged for toxicity, obscenity, insults, and/or identity-based hate. Not all comments are flagged leaving unlabeled outliers. Both datasets have overlapping groups.

Raw text is first converted into term frequency vectors. Preprocessing is performed by the Scikit-learn `TfidfVectorizer` class which deconstructs a set of documents into IDF-weighted term frequency vectors [23]. N-grams of two to four words are included. Terms occurring in at least 0.1% of documents are included.

Neo-k-medoids requires a pairwise distance matrix between all $x \in X$. Computing an $n \times n$ distance matrix requires n^2 computations and becomes computationally expensive if poorly implemented. The Scikit-learn `sklearn.metrics.pairwise_distances` function implements efficient calculations of popular distance and dissimilarity metrics [23]. Distance matrices are generated for seven of these metrics.

The "correct" values for k, α and β cannot be known directly from a raw document collection. Values that describe the manual classes are used for testing.

The Neo-k-medoids algorithm is run 250 times per distance matrix per corpus. The best local minimum selected as the final result. NMF and random clusters are also produced as a point of reference. For context, clusters that satisfy k, α, and β are produced with NMF and through random generation.

5. Results

Cluster validation is based on F_1 [15], purity [5], and silhouette scores [24]. Other external scores, such as entropy, penalize clusters that contain objects from multiple classes [6]. This does not make sense in disjunctive clustering, where strong clusters will contain objects from multiple classes. Silhouette score are calculated for both clusters and manual classes. Results are presented in Table 2. High values are bolded.

Initial results are disappointing. Neither NMF nor neo-k-medoids significantly improve purity scores over random clusters. Both algorithms have low F_1 scores for movie data and average F_1 scores for toxic comments.

Interesting results arise regarding silhouette scores. Under certain conditions neo-k-medoids produces clusters with high silhouette scores. This means that, under the given metric, clusters have low within-cluster distance and high between-cluster distance. Manual class silhouette scores are all on par or worse than random clusters. The goal is to emulate manual classes; the low scores suggest that the chosen metrics do not capture key dissimilarities that people focus on. These results suggest that neo-k-medoids can produce strong clusters, but that an alternative dissimilarity measure is needed to align

Table 2. Cluster validation results

Dataset	Algorithm	Metric	F_1	Purity	Cluster Silhouette	Class Silhouette
Movie Plots	NMF		0.23	0.54	-	-
	avg 50 random		0.11	0.52	-0.00	0.00
	neo-k-medoids	chebyshev	0.19	0.52	0.12	-0.04
		cityblock	0.23	0.52	-0.01	-0.03
		cosine	0.19	0.52	0.06	0.00
		euclidean	0.23	0.52	0.01	0.00
		hamming	0.20	0.53	0.17	-0.07
		matching	0.20	0.53	0.17	-0.07
		russellrao	0.20	0.53	0.00	-0.00
Movie Titles	NMF		0.19	0.48	-	-
	avg 50 random		0.11	0.52	-0.03	-0.13
	neo-k-medoids	chebyshev	0.20	0.55	**0.74**	-0.11
		cityblock	0.19	0.56	0.35	-0.17
		cosine	0.20	0.55	**0.52**	-0.00
		euclidean	0.19	0.56	**0.42**	-0.13
		hamming	0.20	0.53	-0.01	-0.20
		matching	0.19	0.58	0.32	-0.20
		russellrao	0.19	0.58	0.00	-0.00
Toxic Comments	NMF		**0.49**	0.95	-	-
	avg 50 random		0.30	0.93	-0.00	0.00
	neo-k-medoids	chebyshev	0.44	0.92	-0.03	-0.02
		cityblock	0.45	0.93	0.00	-0.00
		cosine	0.44	0.92	0.00	0.01
		euclidean	0.45	0.93	-0.00	0.00
		hamming	0.45	0.95	**0.66**	-0.02
		matching	0.45	0.95	-0.06	-0.01
		russellrao	0.45	0.95	0.00	0.00

clusters with manual labels.

Observe that higher silhouette scores align with shorter documents. Of the datasets, movie titles are the shortest and movie plots the longest. One possible cause may be the roll of keywords. Short texts convey meaning with only a few terms. It may be easier to distinguish between texts even if they share only a few words. More testing is required to explore this hypothesis.

Neo-k-medoids failed to emulate the partition of movie genres and toxic Wikipedia comments relative to manually tagged classes. Weak manual class silhouette scores suggest that neo-k-medoids might produce stronger results with an alternate dissimilarity measure. In its current state, the algorithm is unable to produce concept groups similar to those produced by a team of coders.

6. Conclusions

Document coding reveals the valuable information within text, but the process is time consuming and expensive. Coders group text into logical groups based on important concepts. Some documents span multiple topics whereas other outliers discuss nothing of importance. Constructive automation needs to emulate these properties to make text data more approachable.

Popular clustering techniques generate strong results but are either susceptible to outliers or produce disjoint clusters. Little research exists on disjunctive clustering and proposed methods only apply to numeric data. Neo-k-medoids introduces two constraints to the k-medoids objective function to produce non-exhaustive, overlapping clusters. The implemented algorithm converges monotonically towards a local minimum of the

objective function from a random medoid initialization.

Future work should test the proposed algorithm on a variety of tagged text data to better understand performance. In practice, k, α, and β are unknown; a technique is needed to estimate these values from the data. In the current implementation of the proposed algorithm, initial medoid selection is completely random. Possible performance and accuracy gains may arise from more intelligent selection, such as initializing the most dissimilar documents as medoids. Final results depend on medoid initialization and a strong choice may improve results and decrease the required number of trials.

Neo-k-medoids failed to replicate manually assigned classes from movie genres and Wikipedia comments. This failure is a concern but does not immediately invalidate neo-k-medoids. Low manual class silhouette scores suggest that the tested dissimilarity measures do not capture the differences manual coders care about. Future research should explore alternative dissimilarity measures that better capture the properties people use to distinguish text. Neo-k-medoids may produce better partitions when paired with such a measure.

References

[1] J. Saldaña, *The Coding Manual for Qualitative Researchers*, 2nd ed. Sage Publications Ltd, 2013.

[2] D. Card and N. A. Smith, "Automated coding of open-ended survey responses," Ph.D. dissertation, Carnegie Mellon University, October 2015.

[3] U. Reja, K. Manfreda, V. Hlebec, and V. Vehovar, "Open-ended vs. close-ended questions in web questionnaires," *Developments in Applied Statistics*, vol. 19, 01 2003.

[4] A. S. Agrawal and S. Bojewar, "Comparative study of various clustering techniques," *International Journal of Computer Science and Mobile Computing*, vol. 3, pp. 497–504, October 2014.

[5] A. Huang, "Similarity measures for text document clustering," in *Proceedings of the New Zealand Computer Science Research Student Conference*, April 2008.

[6] C. D. Manning and H. Schütze, *Foundations of Statistical Natural Language Processing*. The MIT Press, 1999.

[7] M. P. Naik, H. B. Prajapati, and V. K. Dabhi, "A survey on semantic document clustering," in *2015 IEEE International Conference on Electrical, Computer and Communication Technologies (ICECCT)*, March 2015, pp. 1–10.

[8] W. Xu, X. Liu, and Y. Gong, "Document clustering based on non-negative matrix factorization," NEC Laboratories America, Inc, Tech. Rep., 2003.

[9] R. Mihalcea, C. Corley, and C. Strapparava, "Corpus-based and knowledge-based measures of text semantic similarity," in *Proceedings of the 21st national conference on Artificial intelligence*, vol. 1. AAAI American Association for Artificial Intelligence, July 2016, pp. 775–780.

[10] R. Baghel and R. Dhir, "A frequent concept based document clustering algorithm," *International Journal of Computer Applications*, vol. 4, no. 5, July 2010.

[11] S.-S. Choi, S.-H. Cha, and T. C. C., *Systemics, Cybernetics, and Informatics*, no. 1, pp. 43–48, 2010.

[12] M. Rafi and M. Shahid Shaikh, "An improved semantic similarity measure for document clustering based on topic maps," *CoRR*, 2013. [Online]. Available: http://arxiv.org/abs/1303.4087

[13] "Wordnet: A lexical database for english," 2010, [Online; accessed 5-November-2018]. [Online]. Available: https://wordnet.princeton.edu/

[14] M. Steinbach, G. Karypis, and V. Kumar, "A comparison of document clustering techniques," in *KDD Workshop on Text Mining*, 2000.

[15] J. J. Whang, Y. Hou, D. F. Gleich, and I. S. Dhillon, "Non-exhaustive, overlapping clustering," *IEEE Transactions On Pattern Analysis And Machine Intelligence*, 2018.

[16] J. C. Bezdek, R. Ehrlich, and W. Full, "Fcm: The fuzzy c-means clustering algorithm," *Computers & Geosciences*, vol. 10, no. 2-3, pp. 191–203, 1984.

[17] S. Baadel, F. Thabtah, and J. Lu, "Mcoke: Multi-cluster overlapping k-means extension algorithm," *International Journal of Computer and Information Engineering*, vol. 9, no. 2, 2015.

[18] C. Bauckhage, "Lecture notes on data science: Soft k-means clustering," University of Bonn, Tech. Rep., October 2015.

[19] L. Kaufman and P. J. Rousseeuw, "Clustering by means of medoids," in *Statistical Data Analysis Based on the L1-Norm and Related Methods*. Elsevier Science Pub. Co., 1987, pp. 405–416.

[20] E. Kerstens, "Neo-k-medoids," https://github.com/ekerstens/neokmedoids/tree/master, 2019.

[21] Jigsaw. (2019) Toxic comment classification challenge. [Online]. Available: https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge/data

[22] TheMitchWorksPro. (2019) Imdb top 250 lists and 5000 plus imdb records. [Online]. Available: https://data.world/studentoflife/imdb-top-250-lists-and-5000-or-so-data-records

[23] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[24] O. Arbelaitz, I. Gurrutxaga, J. Muguerza, J. M. Pérez, and I. Perona, "An extensive comparative study of cluster validity indices," *Pattern Recognition*, vol. 46, pp. 243–256, 2013.