Introduction:

k-means clustering is an algorithm that takes a set of data items and divides them into k groups of items that are supposed to be more similar to each other than to the items in the other groups. It is an unsupervised learning algorithm, as it takes unlabeled observations and tries to discover the underlying structure of the data, instead of training a model using a set of labelled data and then using that model to predict the value of the label for a set of test data (which is what is done in supervised learning).

k-means can be used in the field of text mining for text clustering. Text clustering algorithms take a set of text objects, such as sentences, paragraphs, chapters, articles, or websites, and divide them into groups of related objects. Doing this has many applications. For example, when trying to get a sense of what a particular text collection contains, clustering its documents can help you figure out what categories of content are present, so that you can look at a few texts from each category to get a better understanding of the whole corpus. Also, if you are trying to develop a text retrieval or browsing application, text clustering can help remove duplicate items or identify related documents so that they can be displayed together. In addition, if you are trying to run some text analysis algorithm, clustering your documents and marking each one with its cluster number introduces another predictor variable that might improve your results. k-means clustering is a type of similarity-based clustering, in which documents are grouped together due to being more similar to each other than to documents in other groups (as opposed to generative probabilistic methods of clustering, where the characteristics of various groups of documents are estimated and each text object is assigned to the group whose underlying distribution seems most likely to generate it). In the remainder of this technology review, I will describe how k-means clustering is carried out and give an overview of some related algorithms.

Inputs and Preprocessing:

When performing text clustering using k-means, documents will typically be treated as bags of words. Each document is viewed as an unordered list of all the words it contains and will be fed into the algorithm as a vector with one element for each unique term in your collection. Representing the structure of each document more deeply than this is usually unnecessary and counterproductive, though you may sometimes want to add additional terms to each document's vector representation which contain information that might otherwise not be conveyed. However, representing each document by a vector of raw word counts may lead to poor results, so each document's contents must be processed before being fed into the k-means algorithm.

After dividing each document into a list of words, one should remove stopwords (very common words that do not tell you much about documents) from those lists. By doing this, you can make your program run faster and remove irrelevant information that might cause overfitting. You should also perform stemming on the tokens of each document, to convert all words into standardized forms of themselves. If you did not do this, then the words "run," "ran," and "running" would be treated as distinct, even though they all mean essentially the same thing. In addition, instead of representing each document as a raw term frequency vector, in which each element of the vector tells you how many times a specific word shows up in that document, you should perform TF-IDF weighting. This means that you should perform a sublinear transformation on each word's frequency, so adding additional instances of a single word has diminishing returns, and you should make words that appear in fewer documents more important, as those words are likely more useful for distinguishing different groups of documents. You might also want to perform document length normalization, where you divide the vector element for each term by a number corresponding to the length of its document (such as the document

length or the highest term frequency within that document). If you do not do this, calculating Euclidian distances between the vector representations of your documents may reflect the magnitude of the vectors instead of similarity in content. Many of these steps are similar to the ones performed while doing text retrieval. In both cases, carrying out these steps may improve your algorithm's results.

The k-means clustering algorithm may be slow in some cases, especially if you have a lot of documents and terms. Its asymptotic computational complexity is O(i * n * p * k), where i is the number of iterations before convergence, n is the number of documents to be clustered, p is the total number of unique terms across all documents, and k is the number of clusters. Reducing the number of unique terms can help improve runtime, as computing the distance between two points becomes faster when there are fewer terms. One way in which the number of unique terms can be reduced is through PCA, which transforms your data into a lower-dimensional representation which retains most of the information in the original dataset. Another method for accomplishing this is to perform random projection. In random projection, you project the vectors representing your documents onto some sort of randomly generated matrix with less dimensionality (the contents of this matrix vary depending on what type of random projection you use). These dimensionality reduction methods may reduce the quality of your clustering results, but the runtime improvements they yield may sometimes be worthwhile.

Algorithm:

The basic algorithm behind k-means clustering is fairly simple. First, you randomly choose k points, which will be your initial guesses for the centers of your k clusters. Secondly, you iterate through all the data points, and assign each one to the cluster whose center it is closest to. Various methods for measuring distance can be used, but Euclidean distance between the vector representations of text objects and cluster centers is a common method used when performing text clustering using k-means. Thirdly, for each cluster, you set the cluster center to be the point that has the minimum distance to all observations in the cluster (this point does not have to be an observation). When performing text clustering using k-means, the new center for each cluster is set to be the mean of all the text objects in that cluster. Finally, you repeat steps two and three until convergence. You can tell that the algorithm has converged and stop running it when the total amount by which the cluster centers has changed or the decrease in some cost function (perhaps inertia, which will be defined later) drops below some threshold. This algorithm is analogous to the EM-algorithm, with step two being equivalent to the E-step and step three being equivalent to the M-step. However, instead of finding the probability that each observation belongs to each cluster, we assign the observations to exactly one cluster each.

Each iteration of the k-means clustering algorithm improves the quality of the clustering (in terms how similar the items within each cluster are, relative to the difference between clusters), and the algorithm is guaranteed to converge to a state where the clusters stop changing. However, it may converge to a local minimum instead of a global minimum, meaning that another assignment of observations to clusters may be better. To deal with this, you should run the k-means algorithm multiple times, with a different set of randomly generated initial cluster centers each time, and choose the best clustering.

Number of Clusters to Use:

There are a variety of methods for choosing k, the number of clusters to classify observations into. One way of doing this is called the elbow method. To understand the elbow method, you must know what inertia and distortion are. Inertia is the total distance between each observation and its cluster center, and distortion is the average distance between each observation and its cluster center. The lower these are, the better. However, they always decrease when you increase the number of clusters, so we wish to pick a value of k that results in a low value of inertia or distortion yet is small enough to have meaningfully sized clusters. To accomplish this, you perform k-means clustering with many values of k, then make a plot of the results, with the values of k on the x-axis and the inertia or distortion obtained from performing k-means clustering using each value of k on the y-axis. You should then select the value of k where the decrease in distortion or inertia starts to slow down significantly.

Other methods of selecting the ideal number of clusters involve calculating some statistics about the results of your clustering for different values of k, and then using the values of that statistic for various numbers of clusters to find the best number of clusters. One such statistic is the silhouette coefficient, which measures how far each point is from its own cluster center, relative to how far it is from the second closest cluster center. You can cluster your documents with various values of k and select either the value of k that results in the largest silhouette coefficient or the smallest value of k that results in a silhouette coefficient above some threshold (such as 0.5, which indicates that some structure has been found in the data, or 0.7, which indicates that a strong structure has been found in the data). There are many other methods you can use to determine the best number of clusters, but the elbow method and calculating silhouette statistics for various values of k are two common ones.

Evaluation:

Evaluating the effectiveness of an unsupervised learning algorithm such as k-means is harder than evaluating the effectiveness of a supervised learning algorithm. When evaluating a supervised learning algorithm, you can just take the data that was not used for training your model, use your model to either predict either a class label or a numerical value for each test example, and see how close your predictions were. For k-means clustering, there are no labels on the data, and there is no canonical way in which the observations should be split. Two main methods have been developed to evaluate the effectiveness of text clustering algorithms like k-means. One method is direct evaluation, in which the assignment of documents to clusters by k-means is compared to the result of a manual assignment of documents to clusters by humans. This method is rather labor intensive. Another method is indirect evaluation, which is only applicable when the results of your clustering are going to be used in some application with quantifiable results (for instance, you will use the cluster labels as one predictor in an algorithm to classify emails as “spam” or “not spam”). You record how effective the target application is when given unclustered data, then see if clustering the data helps that application performs more effectively. If it does, then the clustering was successful.

Variants of k-Means Clustering:

There are several clustering algorithms that are very similar to k-means. One is k-medians clustering, which is the same as k-means, but when assigning observations to the cluster whose center is nearest with regards to Euclidean (L2) distance, you instead assign them to the cluster whose center is nearest with regards to Manhattan (L1) distance, and when computing the new cluster centers, you set the cluster center's vector to be the element-wise median of all the observation vectors within that cluster. Also, you should use Manhattan distances when calculating distances for other reasons (such as when you are trying to figure out the best value of k, or when to stop the algorithm, or which set of clustering results from your multiple initializations is the best). k-medians clustering may lead to better results than k-means in some cases, as it is more resistant to outlier documents that are very different from the rest. The intuition is that if you have the numbers 1, 2, 3, 4, 5, 6, 7, 8 and 1000, then the mean is 115.1111, which is far from most of the data, but the median is 5, which is close to most of the observations.

Another similar algorithm is k- medoids clustering. In k-medoids clustering, often implemented partition around medoids (PAM), each cluster center must be one of your observations. It is very useful in cases where computing the mean of multiple observations is impossible. For text clustering, this is not a problem, but k-medoids may still be useful, as it provides greater resistance against outliers than k-means (because extreme values are less capable of affecting cluster centers when those centers are required to be observations). To perform k-medoids clustering, you choose k random points as cluster centers, assign each point to the closest cluster center, then try swapping non-medoid observations with the medoids (so that the non-medoid becomes the cluster center and the medoid stops being the cluster center) and making any swaps that decrease your cost function (perhaps total distance between each

observation and its cluster center) until the cluster centers stop moving.

A third related algorithm is kernel k-means clustering. Regular k-means clustering is most suited for cases where the clusters in the data are roughly spherical, of similar sizes, do not overlap, and are linearly separable, and where there are few outliers, and it may yield poor results if some of these conditions are violated. In these cases, using kernel k-means may create a better clustering. The algorithm used is mostly the same, except when computing distances between points, you use kernel functions to compute their distance in a higher dimensional space, without actually projecting them into that space. By doing this, you can create a good partition of your data even when the clusters are not spherical, vary in size, overlap, or are not linearly separable, and when outliers are present. However, kernel k-means is somewhat slower than regular k-means clustering.

Conclusion:

In conclusion, k-means clustering is an unsupervised learning algorithm that divides data entries into a predefined number of groups. It is versatile and can be used in many different areas, including in text mining, where it can be used to cluster text objects. This can help accomplish many tasks, such as aiding people in understanding a collection of documents better, allowing various search and browsing applications to display information more effectively, and boosting the accuracy of other text mining algorithms. Many libraries in many different programming languages offer implementations of the k-means algorithm, as well as functions to help you preprocess the text data you want to cluster and choose a good number of clusters to use. Numerous variants of the k-means algorithm exist, with k-medians, k-medoids, and kernel k-means being among them. Trying them could be a good option if regular k-means clustering does not achieve the results you want. While this technology review has not covered every detail about the theory, implementation, applications, and variants of k-means clustering, it should serve as a basic overview of the algorithm which is sufficient for anyone looking for a good method to cluster text objects.

Works Consulted:

"6.6. Random Projection." Scikit Learn, https://scikit-learn.org/stable/modules/random_projection.html.

Arvai, Kevin. "K-Means Clustering in Python: A Practical Guide." Real Python, 2020, https://realpython.com/k-means-clustering-python/.

Bester, Tristan. "K-Means And K-Medians." Machine Learning Journey, 27 Feb. 2020, https://machinelearningjourney.com/index.php/2020/02/07/k-means-k-medians/.

de Sá, Lucas. "Text Clustering with K-Means." Medium, 17 Dec. 2019, https://medium.com/@lucasdesa/text-clustering-with-k-means-a039d84a941b.

"K-Medoids Clustering with Solved Example." Geeks for Geeks, 31 Aug. 2022, https://www.geeksforgeeks.org/ml-k-medoids-clustering-with-example/.

Kaushiva, Mahima. "Clustering with K-Means." Towards Data Science, 2 Jan. 2020, https://towardsdatascience.com/clustering-with-k-means-1e07a8bfb7ca.

"Kernel k-Means Clustering Algorithm." Data Clustering Algorithms, 30 Aug. 2020, https://sites.google.com/site/dataclusteringalgorithms/kernel-k-means-clustering-algorithm.

Koch, Korbinian. "A Friendly Introduction to Text Clustering." Towards Data Science, 25 Mar. 2020, https://towardsdatascience.com/a-friendly-introduction-to-text-clustering-fa996bcefd04.

Tutar, Meltem. "Understanding K-Means Clustering and Kernel Methods." Medium, 8 Sept. 2021, https://medium.com/udemy-engineering/understanding-k-means-clustering-and-kernel-methods-afad4eec3c11.

I also used information learned in this class and previous machine learning courses.