Lab 5

Name: Srivenkatesh Nair
Student ID: 378000329
Username: sn6711
ISTE.612

Task 1:Pre-process documents to construct VSM representations

```java
// Method to preprocess documents and create document vectors
public void preprocess(String[] docs) {
    // Create a list of unique terms (vocabulary)
    Set<String> vocabSet = new HashSet<String>();
    for (String doc : docs) {
        String[] terms = doc.split(regex:" ");
        vocabSet.addAll(Arrays.asList(terms));
    }
    ArrayList<String> vocabList = new ArrayList<String>(vocabSet);
    Collections.sort(vocabList);
    // System.out.println(vocabList);

    // Create Doc objects for each document
    for (String doc : docs) {
        this.documents.add(new Doc(doc, vocabList));
    }

}
```

```java
/**
 *
 * Document class for the vector representation of a document
 */
class Doc {
    public ArrayList<Double> termFreq;  // termFreq: Stores the frequency of each term
    public ArrayList<String> termIndex;  // termIndex: Stores the list of terms (vocabulary)

    // Constructor to create a document vector from text
    public Doc(String text, ArrayList<String> allTerms) {
        this.termIndex = allTerms;
        this.termFreq = new ArrayList<Double>(Collections.nCopies(termIndex.size(), o:0.0)); // Initialize with 0s

        String[] terms = text.split(regex:" ");

        for (String term : terms) {
            int index = termIndex.indexOf(term);
            if (index != -1) {
                termFreq.set(index, termFreq.get(index) + 1);
            }
        }
    }

    // Copy constructor
    public Doc(Doc doc) {
        this.termFreq = new ArrayList<>(doc.termFreq);
        this.termIndex = doc.termIndex;
    }

    @Override
    public String toString() {
        return termFreq.toString();
    }
}
```

Task 2: Cluster documents

```java
// Method to perform k-means clustering
public void cluster() {
    // Choose initial centroids (first and ninth documents)
    Doc centroid1 = new Doc(documents.get(index:0));
    Doc centroid2 = new Doc(documents.get(index:8));
    ArrayList<Doc> initialCentroids = new ArrayList<Doc>();
    initialCentroids.add(centroid1);
    initialCentroids.add(centroid2);

    ArrayList<Double> temp = new ArrayList<Double>();
    ArrayList<ArrayList<Double>> newCentroids = new ArrayList<ArrayList<Double>>();
    newCentroids.add(temp);
    newCentroids.add(temp);



    boolean changed = true;
    boolean flag = true;

    while (changed == true) {
        // Clear previous clusters
        for (ArrayList<Doc> cluster : clusters) {
            cluster.clear();
        }

        // Assign each document to the nearest centroid
        for (Doc doc : documents) {
            double minDistance = Double.MAX_VALUE;
            int closestCentroidIndex = -1;
            if (flag == true){
                for (int i = 0; i < initialCentroids.size(); i++) {
                    double distance = distanceTo(initialCentroids.get(i).termFreq, doc.termFreq);
                    if (distance < minDistance) {
                        minDistance = distance;
                        closestCentroidIndex = i;
                    }
                }
```

```
Cluster: 0
0  1  2  3

Cluster: 1
4  5  6  7  8  9
```