```
import java.io.IOException;

import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;

public class dedupMapper extends Mapper<LongWritable,Text,Text,Text>
{

	@Override
	public void map(LongWritable key, Text value, Context context)
	{
		Text second = new Text("Dummy data");

		try
	    {
			String str[] = value.toString().split("###");
			if (str.length > 1)
			{
				// first = new Text(str[0]);
				second = new Text(str[1]);
			}

			context.write(second, second);
```

```
            }
                catch (IOException | InterruptedException e)
                {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                }

        }
        public void run(Context context) throws IOException, InterruptedException
        {
                setup(context);
                while (context.nextKeyValue())
                {
                        map(context.getCurrentKey(), context.getCurrentValue(), context);
                }
                cleanup(context);
        }
}



import java.io.IOException;


import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;
```

```
public class dedupMapper extends Mapper<LongWritable,Text,Text,Text>
{

	@Override
	public void map(LongWritable key, Text value, Context context)
	{
		Text second = new Text("Dummy data");


		try
	    {
			String str[] = value.toString().split("###");
			if (str.length > 1)
			{
				// first = new Text(str[0]);
				second = new Text(str[1]);
			}


			context.write(second, second);
	    }
		catch (IOException | InterruptedException e)
		{
			// TODO Auto-generated catch block
			e.printStackTrace();
		}

	}
```

```
        public void run(Context context) throws IOException, InterruptedException
        {
                setup(context);
                while (context.nextKeyValue())
                {
                        map(context.getCurrentKey(), context.getCurrentValue(), context);
                }
                cleanup(context);
        }
}
```

```
import org.apache.hadoop.io.Text;
import org.apache.hadoop.io.WritableComparable;
import org.apache.hadoop.io.WritableComparator;

public class groupingComparator1 extends WritableComparator
{

        public groupingComparator1()
        {
           super(Text.class, true);
      }
        @SuppressWarnings("rawtypes")
        @Override
        public int compare(WritableComparable w1, WritableComparable w2)
```

```
	{
		Text t1= (Text) w1;
		Text t2= (Text) w2;
		String str1=t1.toString();
		String str2=t2.toString();
		//str1.compareTo(str2);
		System.out.println("The two strings which are getting compared are"+ str1+" and
"+str2);
		Double result=similarity(str1,str2)*100;
		if(result>=70.00)
		{
			System.out.println("The two strings matched are"+ str1+" and "+str2);
			return 0;

		}
		else if(str1.length()>str2.length())
			return 1;
		else
			return -1;

	}

	public double similarity(String s1, String s2)
	{
	    String longer = s1, shorter = s2;
	    if (s1.length() < s2.length())
```

```
    { // longer should always have greater length


        longer = s2; shorter = s1;
    }
    int longerLength = longer.length();
    if (longerLength == 0) { return 1.0; /* both strings are zero length */
    }


    return (longerLength - editDistance(longer, shorter)) / (double) longerLength;


  }
public    int editDistance(String s1, String s2)
{
                s1 = s1.toLowerCase();
                s2 = s2.toLowerCase();


                int[] costs = new int[s2.length() + 1];
                for (int i = 0; i <= s1.length(); i++)
                {
                    int lastValue = i;
                    for (int j = 0; j <= s2.length(); j++)
                    {
                        if (i == 0)
                            costs[j] = j;
                        else
                        {
```

```
                    if (j > 0)
                    {
                        int newValue = costs[j - 1];
                        if (s1.charAt(i - 1) != s2.charAt(j - 1))
                            newValue = Math.min(Math.min(newValue, lastValue),
                                    costs[j]) + 1;
                        costs[j - 1] = lastValue;
                        lastValue = newValue;
                    }
                }
            }
            if (i > 0)
                costs[s2.length()] = lastValue;
        }
        return costs[s2.length()];
    }


}


import java.io.IOException;
import java.util.ArrayList;

import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;
```

```
public class dedupReducer extends Reducer<Text,Text,Text,Text>

{


        @Override

         public void reduce(Text key, Iterable<Text> value,Context context) throws IOException,
InterruptedException

         {


                ArrayList<String> ar= new ArrayList<String>();

                        for(Text t1:value)

                        {

                                ar.add(t1.toString());

                        }

                        /*if(ar.size()>1)

                        {*/

                          try{

                                  context.write(key, new Text(ar.toString()));

                          }

                catch(Exception e)

                {

                          e.printStackTrace();

                }
```

```
                        //}
        }

        @Override
        public void run(Context context) throws IOException, InterruptedException
        {
                        setup(context);
                        while (context.nextKey())
                        {
                                reduce(context.getCurrentKey(), context.getValues(), context);
                        }
                        cleanup(context);
        }
}
```