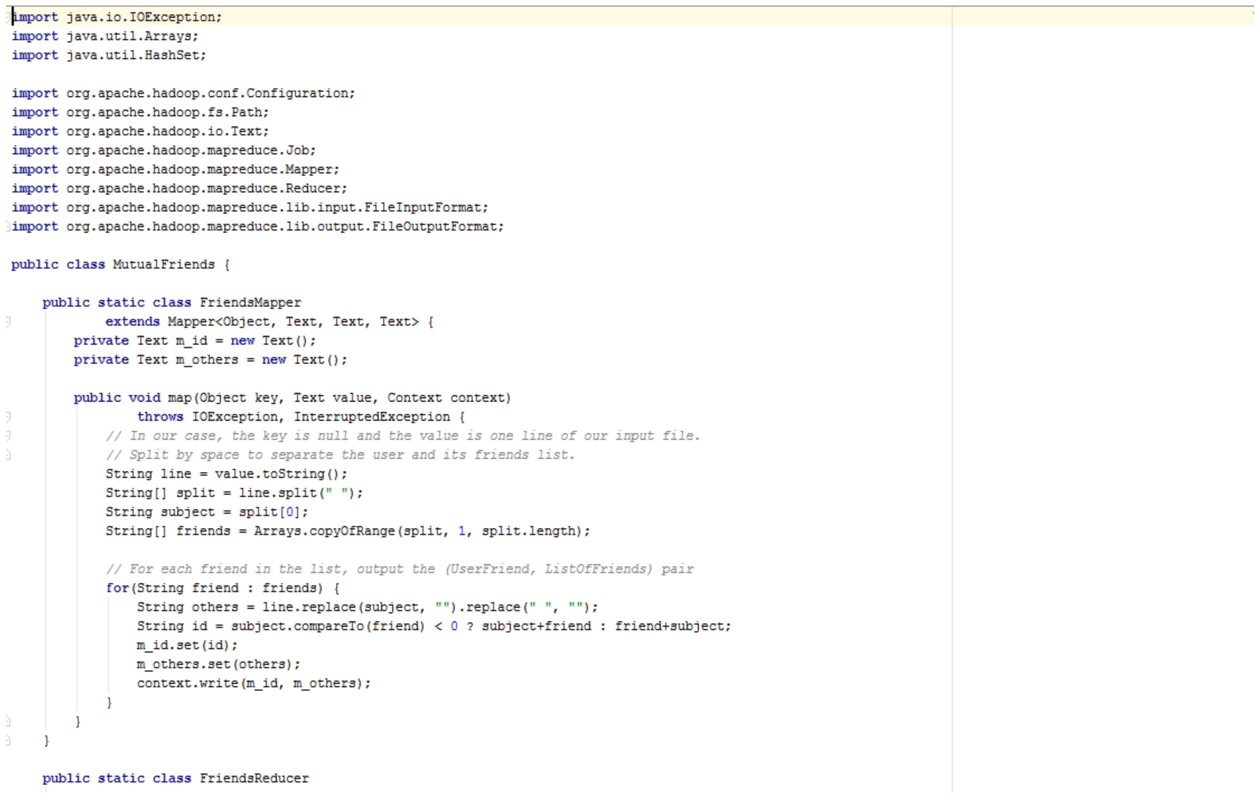
Report for Lab Assignment 4

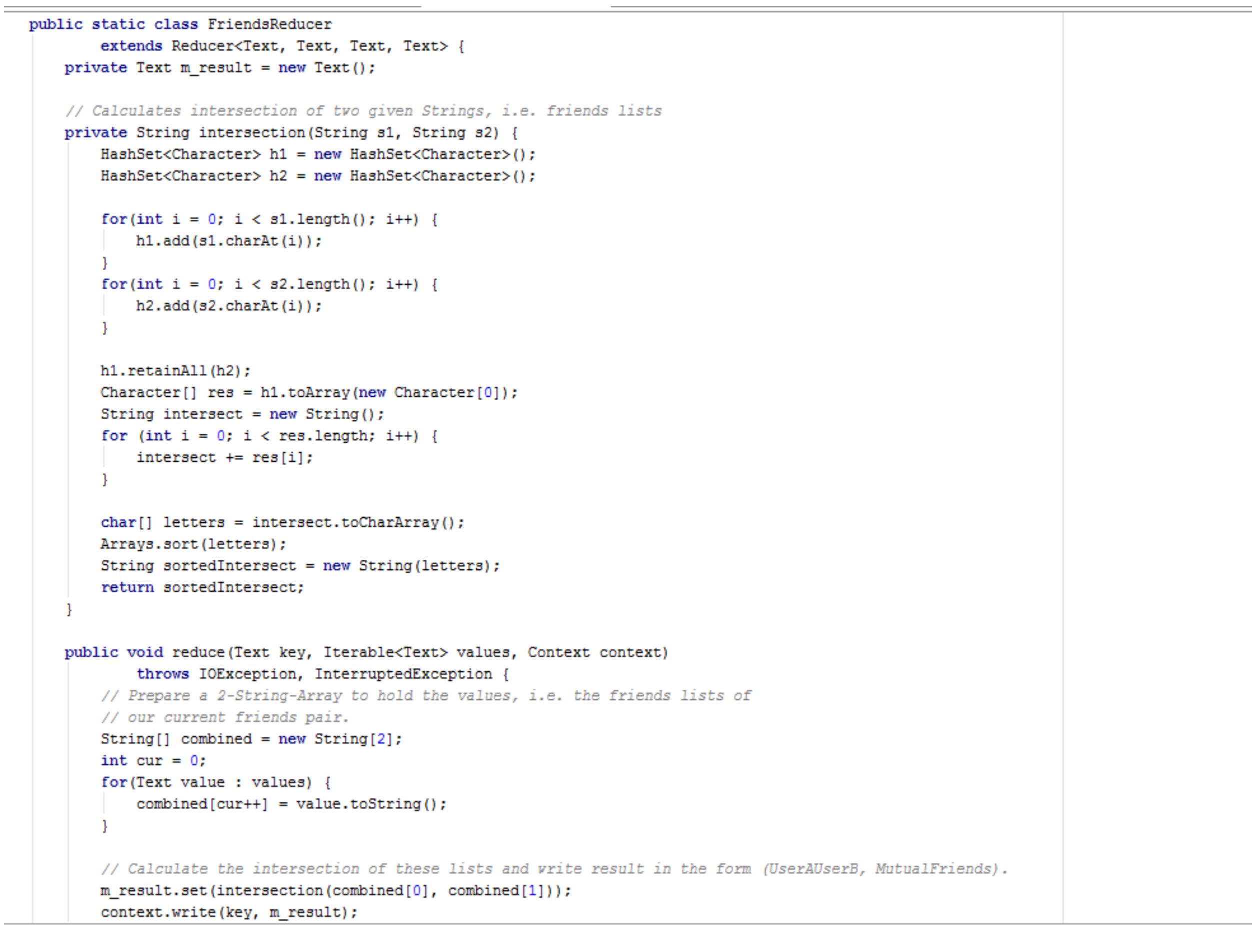
1. Hadoop MapReduce Algorithm

Implement MapReduce algorithm for finding Facebook common friends problem and run MapReduce job on Apache Hadoop. Write a report including your algorithm and result screenshots.

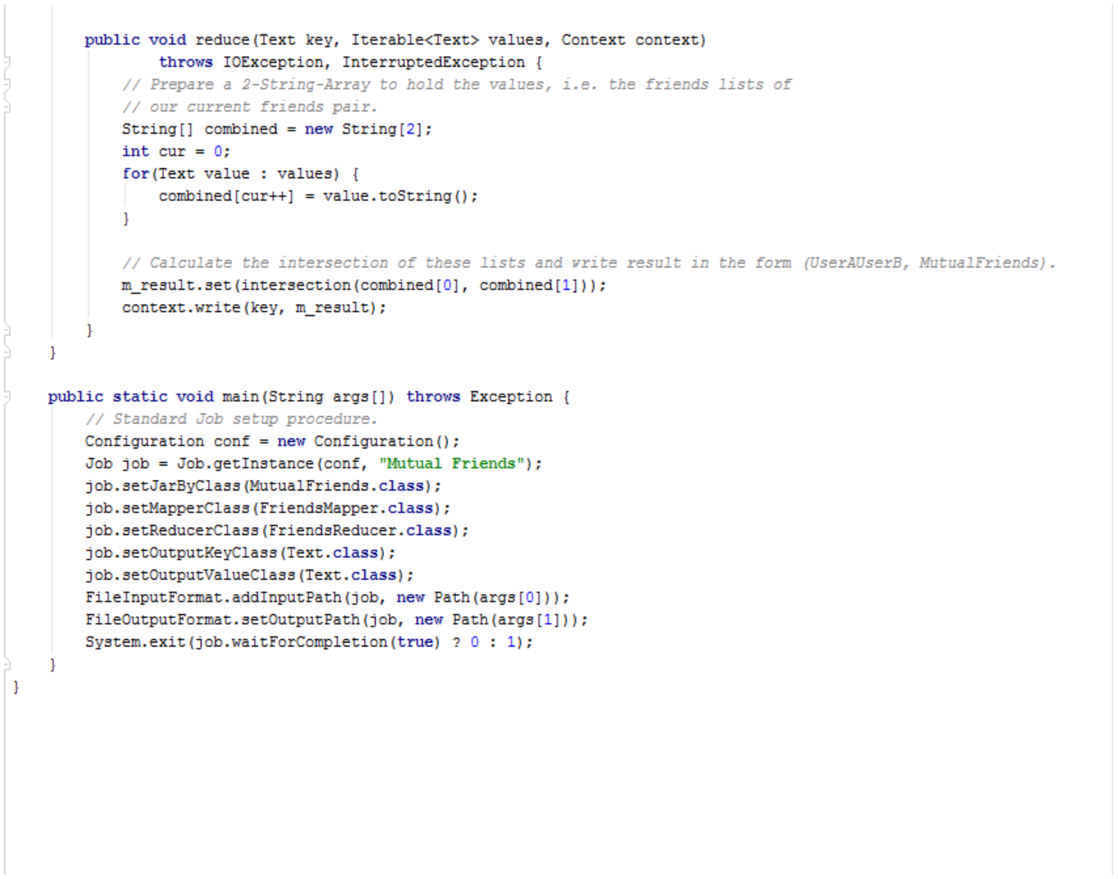
Description: The following code, taken from Github, solves the particular problem.



```
import java.io.IOException;
import java.util.Arrays;
import java.util.HashSet;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

public class MutualFriends {

    public static class FriendsMapper
            extends Mapper<Object, Text, Text, Text> {
        private Text m_id = new Text();
        private Text m_others = new Text();

        public void map(Object key, Text value, Context context)
                throws IOException, InterruptedException {
            // In our case, the key is null and the value is one line of our input file.
            // Split by space to separate the user and its friends list.
            String line = value.toString();
            String[] split = line.split(" ");
            String subject = split[0];
            String[] friends = Arrays.copyOfRange(split, 1, split.length);

            // For each friend in the list, output the (UserFriend, ListOfFriends) pair
            for(String friend : friends) {
                String others = line.replace(subject, "").replace(" ", "");
                String id = subject.compareTo(friend) < 0 ? subject+friend : friend+subject;
                m_id.set(id);
                m_others.set(others);
                context.write(m_id, m_others);
            }
        }
    }

    public static class FriendsReducer
```



```java
public static class FriendsReducer
        extends Reducer<Text, Text, Text, Text> {
    private Text m_result = new Text();

    // Calculates intersection of two given Strings, i.e. friends lists
    private String intersection(String s1, String s2) {
        HashSet<Character> h1 = new HashSet<Character>();
        HashSet<Character> h2 = new HashSet<Character>();

        for(int i = 0; i < s1.length(); i++) {
            h1.add(s1.charAt(i));
        }
        for(int i = 0; i < s2.length(); i++) {
            h2.add(s2.charAt(i));
        }

        h1.retainAll(h2);
        Character[] res = h1.toArray(new Character[0]);
        String intersect = new String();
        for (int i = 0; i < res.length; i++) {
            intersect += res[i];
        }

        char[] letters = intersect.toCharArray();
        Arrays.sort(letters);
        String sortedIntersect = new String(letters);
        return sortedIntersect;
    }

    public void reduce(Text key, Iterable<Text> values, Context context)
            throws IOException, InterruptedException {
        // Prepare a 2-String-Array to hold the values, i.e. the friends lists of
        // our current friends pair.
        String[] combined = new String[2];
        int cur = 0;
        for(Text value : values) {
            combined[cur++] = value.toString();
        }

        // Calculate the intersection of these lists and write result in the form (UserAUserB, MutualFriends).
        m_result.set(intersection(combined[0], combined[1]));
        context.write(key, m_result);
```

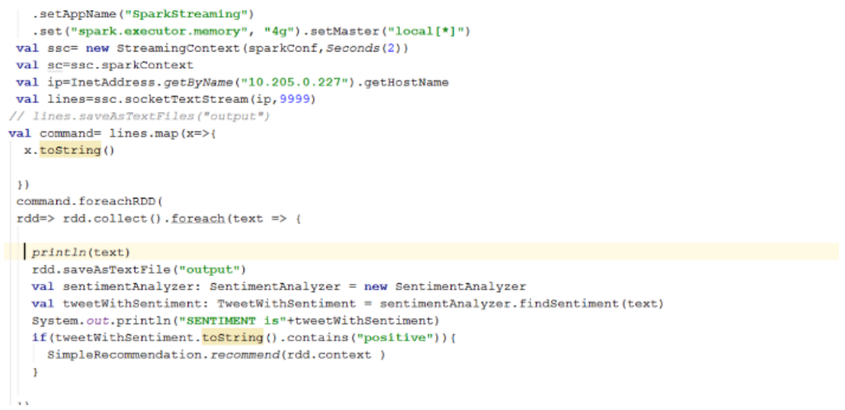


```
        public void reduce(Text key, Iterable<Text> values, Context context)
                throws IOException, InterruptedException {
            // Prepare a 2-String-Array to hold the values, i.e. the friends lists of
            // our current friends pair.
            String[] combined = new String[2];
            int cur = 0;
            for(Text value : values) {
                combined[cur++] = value.toString();
            }

            // Calculate the intersection of these lists and write result in the form (UserAUserB, MutualFriends).
            m_result.set(intersection(combined[0], combined[1]));
            context.write(key, m_result);
        }
    }

    public static void main(String args[]) throws Exception {
        // Standard Job setup procedure.
        Configuration conf = new Configuration();
        Job job = Job.getInstance(conf, "Mutual Friends");
        job.setJarByClass(MutualFriends.class);
        job.setMapperClass(FriendsMapper.class);
        job.setReducerClass(FriendsReducer.class);
        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(Text.class);
        FileInputFormat.addInputPath(job, new Path(args[0]));
        FileOutputFormat.setOutputPath(job, new Path(args[1]));
        System.exit(job.waitForCompletion(true) ? 0 : 1);
    }
}
```

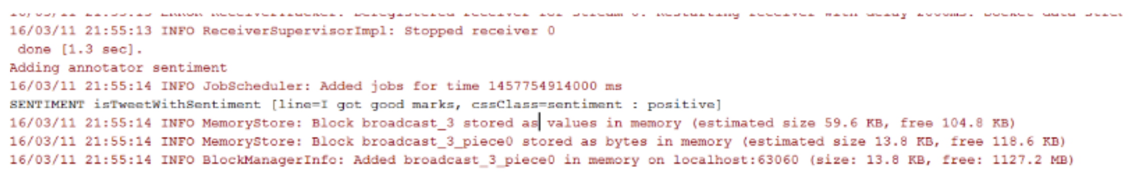
2. Smartphone/Watch Application

Implement a smartwatch/smartphone application using existing speech services/image services (e.g., IBM Alchemyapi, Face++) related to your project.

Description: In our project, we use sentiment analysis to get an idea for speech recognition. Below is an implementation of such Sentiment Analysis.



```
    .setAppName("SparkStreaming")
    .set("spark.executor.memory", "4g").setMaster("local[*]")
  val ssc= new StreamingContext(sparkConf,Seconds(2))
  val sc=ssc.sparkContext
  val ip=InetAddress.getByName("10.205.0.227").getHostName
  val lines=ssc.socketTextStream(ip,9999)
// lines.saveAsTextFiles("output")
val command= lines.map(x=>{
  x.toString()

 })
 command.foreachRDD(
 rdd=> rdd.collect().foreach(text => {

   println(text)
   rdd.saveAsTextFile("output")
   val sentimentAnalyzer: SentimentAnalyzer = new SentimentAnalyzer
   val tweetWithSentiment: TweetWithSentiment = sentimentAnalyzer.findSentiment(text)
   System.out.println("SENTIMENT is"+tweetWithSentiment)
   if(tweetWithSentiment.toString().contains("positive")){
     SimpleRecommendation.recommend(rdd.context )
   }
```



```
[illegible]
16/03/11 21:55:13 INFO ReceiverSupervisorImpl: Stopped receiver 0
 done [1.3 sec].
Adding annotator sentiment
16/03/11 21:55:14 INFO JobScheduler: Added jobs for time 1457754914000 ms
SENTIMENT isTweetWithSentiment [line=I got good marks, cssClass=sentiment : positive]
16/03/11 21:55:14 INFO MemoryStore: Block broadcast_3 stored as values in memory (estimated size 59.6 KB, free 104.8 KB)
16/03/11 21:55:14 INFO MemoryStore: Block broadcast_3_piece0 stored as bytes in memory (estimated size 13.8 KB, free 118.6 KB)
16/03/11 21:55:14 INFO BlockManagerInfo: Added broadcast_3_piece0 in memory on localhost:63060 (size: 13.8 KB, free: 1127.2 MB)
```