Documentation for Assignment 1—MapReduce

We have already gained the example of MapReduce implementation for WordCount. For our assignment 1 we can use the methods provided by the example and do some modification. According to the original example of word count, in order to implement ngram, firstly, we just need to modify the StringTokenizer to let ngrams be the input keys for mapper. Secondly, we have to modify the reducer to append the filenames containing input key to the end of reducer output. We know that the mapper has four parameters. They are input key/value pairs and output key/value pairs. However, in this case, I change the fourth parameter which is used to be Intwritable to Text.

```
extends Mapper<Object, Text, Text, Text> {
```

Why I do this operation is because I want mapper output key and filename so that after shuffling and sorting the reducer will receive key/list<filename,filename…> pairs. Therefore, I also change the parameters of reducer to four Text types.

```
extends Reducer<Text,Text,Text,Text>
```

The flow of my code is as follow:
1.The mapper receive ***Object fileID/Text fileContent*** pairs.
2.Get the ***fileContent*** and store them in a String array.

```
FileSplit fileSplit = (FileSplit)context.getInputSplit();
String currFile = fileSplit.getPath().getName();    /*Name of
String[] words = value.toString().split(" ");       /*String
```

3.According to miscount, set number of words in one ngram.
4.Convert them to ***Text ngram/Text filename*** pairs and send to shuffle stage.

```
for (int i = 0; i < words.length-(ngram-1); i++) {
    String temp = words[i];                          /
    for (int j = i+1; j < (i+ngram); j++) {
        temp += " ";                                 /
        temp += words[j];
    }
    word.set(temp);                          /*Assign
    Text t = new Text(currFile);             /*Change

    /*
     * Key-value pair in form of (word, "filename")
     */
    context.write(word, t);
```

5.Reducer receive ***Text ngram/Text list<filenames,…>*** pairs.
6.Count filenames in ***Text list<filenames,…>*** and count the number of occurrence of a particular ngram.

```
int cnt = 0;
ArrayList<String> fileslist = new ArrayList<String>();
for (Text fil : files) {
    fileslist.add(fil.toString());
}
cnt = fileslist.size();
```

7.Remove the redundant filenames in list<filenames…>.
8.Output the ***Text ngram/Text # of ngram occurrence+filenames*** pairs.

```
HashSet<String> h = new HashSet<String>(fileslist);
fileslist.clear();
fileslist.addAll(h);
Collections.sort(fileslist);                        /*Sort filenames*/
for (int i = 0; i < fileslist.size(); i++) {
    textresult += " " + fileslist.get(i);
}
Text t = new Text(textresult);
```