Assignment 7

In this function we will use the concept of User Defined Functions for enhanced word count problem.

```
REGISTER 'hdfs://localhost/user/cloudera/pig/counter.jar';
DEFINE C0 skip.counter0();
DEFINE C1 skip.counter1();
DEFINE C3 skip.final_count();
A1 = load 'hdfs://localhost/user/cloudera/pig/UN.txt' as (line:chararray);
A2 = foreach A1 generate TOKENIZE(line) as tokens;
A3 = foreach A2 generate flatten(tokens) as words;
A = foreach A3 generate LOWER(words) as low;
B = load 'hdfs://localhost/user/cloudera/pig/skip.txt' as (skip:chararray);
aa = foreach A generate C0(low) as tuples1;
bb = foreach B generate C1(skip) as tuples2;
A_Final = foreach aa generate $0.$0,$0.$1;
B_Final = foreach bb generate $0.$0,$0.$1;
J1 = UNION A_Final, B_Final;
J2 = foreach J1 generate (chararray)$0 as word, (int)$1 as index;
GRP = group J2 by word;
fin1 = foreach GRP generate C3() as tuples;
final = foreach fin1 generate (chararray)$0.$0 as word, (int)$0.$1 as count;
final = order final by word;
dump final;
store final into 'hdfs://localhost/user/cloudera/pig/results/Enhanced_word_count';
```

1) First Jar file is registered containing all our UDFs.
2) *Package.class()* are defined for easy access.
3) Data is loaded in both relations finally in the form of (word) in A and B.
4) These A and B are passed into defined functions to give output in the form of {(word,0)} and {(word,1)}.
5) A_Final and B_Final remove the nesting and give output in the form of (word,0) and (word,1)
6) UNION of above outputs is done and grouped according to word.
7) Schema for this group is defined and sent into third UDF.
8) Finally output is grouped alphabetically and dumped/saved.

```
2014-02-03 21:54:11,250 [main] I
2014-02-03 21:54:11,272 [main] I
2014-02-03 21:54:11,272 [main] I
(.,1)
(193.,1)
(1945,2)
(1945;,1)
(1947.,1)
(1960s,1)
(1970s,1)
(1994,1)
(2001,1)
(2007.,1)
(24,2)
(51,1)
(;,5)
(a,4)
(actions,1)
(after,1)
(agencies,3)
(agency,1)
(aid,1)
(allies,1)
(also,1)
(an,3)
(another,1)
(approving,1)
(april june,1)
(are,1)
(armed,1)
(as,2)
```

UDFs written in JAVA

counter0 class

```
package skip;

importjava.io.IOException;
importorg.apache.pig.EvalFunc;
importorg.apache.pig.data.Tuple;
importorg.apache.pig.data.TupleFactory;


public class counter0 extends EvalFunc<Tuple> {

		public Tuple exec(Tuple input) throws IOException {

		int i =0;
		Tuple out = TupleFactory.getInstance().newTuple(2);

		out.set(0, input.get(0));
		out.set(1,i);

		return out;}}
```

For implementing any evaluation functions they need to extend EvalFunc class and implement exec() method.
Evaluation functions give one output for a particular input in foreach statements.

Characteristics :

1) Input and output both are Tuples.
2) TupleFactory generates an instance of new Tuple. In our case we have defined a tuple with two fields.
3) Input words from UN.txt are added to field0 of output tuple and counter of int 1 is added to field1.

counter1 class

```
package skip;

importjava.io.IOException;

importorg.apache.pig.EvalFunc;
importorg.apache.pig.data.Tuple;
importorg.apache.pig.data.TupleFactory;

public class counter1 extends EvalFunc<Tuple> {
	public Tuple exec(Tuple input) throws IOException {

		int i =1;
		Tuple out = TupleFactory.getInstance().newTuple(2);

		out.set(0, input.get(0));
		out.set(1,i);

		return out;
	}
}
```

Characteristics :

4) Input and output both are Tuples.
5) TupleFactory generates an instance of new Tuple. In our case we have defined a tuple with two fields.
6) Input words from UN.txt are added to field0 of output tuple and counter of int 1 is added to field1.

Method Explanation for Enhanced word count

1) From UN.txt , tuples in the form of (word,0) are generated while from skip.txt, tuples in the form of (word,1) are generated. Counter 0/1 signify their file origin.
2) Out put from both UDFs are cojoined(not JOINED).
   CO-Joined concatenates data into 1.
3) Finally concatenated data is first grouped and then sent to our third UDF.
4) IT is important to remember format of group to be sent into third UDF.
5) Output from a group is in a form of tuple. Second element of this tuple has a bag which contains all tuples with same word.
   Second element of these tuples with same word is either 0 or 1.
6) In third UDF, number of tuples in the bag is counted to give word count. Also a track is kept on second element of these tuples. If in the process we encounter integer 1, output is chararray "skipped".
7) If all encountered integers are 0,(meaning not present in skip.txt) returned tuple has (word,count).

final_count class

```
package skip;

importjava.io.IOException;
importjava.util.Iterator;
importorg.apache.pig.EvalFunc;
importorg.apache.pig.data.DataBag;
importorg.apache.pig.data.Tuple;
importorg.apache.pig.data.TupleFactory;


public class final_count extends EvalFunc<Tuple> {
        public Tuple exec(Tuple input) throws IOException {
                Tuple out = TupleFactory.getInstance().newTuple(2);
                Tuple temp = TupleFactory.getInstance().newTuple(2);


                int count = 0;
                String s = "skipped";
                int index=1;
                DataBagdb = (DataBag)input.get(1);
                Iterator<Tuple> it = db.iterator();


                while(it.hasNext()){
```

DataBag is taken out present in second field of input Tuple.

Iterator is generated to iterate between tuples of this bag.

```
                count=count+1;
                temp=it.next();
                if(Integer.parseInt(temp.get(1).toString())==1){
                        out.set(0,s);
                        out.set(1,index);
                        return out;
                }
        }
        out.set(0,temp.get(0));
        out.set(1,count);

        return out;
    }
}
```

If '1' is encountered in second field of extracted tuple in the middle of count,(skipped ,1) is returned in between.

If only '0' is encountered , (word,count) is returned

```
-----------------------------------------
| A_Final     | bytearray     | bytearray |
-----------------------------------------
|             | the           | 0         |
|             | organisation  | 0         |
|             | won           | 0         |
|             | the           | 0         |
|             | 2001          | 0         |
|             | nobel         | 0         |
|             | peace         | 0         |
|             | prize         | 0         |
|             | and           | 0         |
|             | a             | 0         |
|             | number        | 0         |
|             | of            | 0         |
|             | its           | 0         |
|             | officers      | 0         |
|             | and           | 0         |
|             | agencies      | 0         |
|             | have          | 0         |
|             | also          | 0         |
|             | been          | 0         |
|             | awarded       | 0         |
|             | the           | 0         |
|             | prize.        | 0         |
|             | other         | 0         |
|             | evaluations   | 0         |
|             | of            | 0         |
|             | the           | 0         |
```

```
-------------------------
| A     | low: chararray |
-------------------------
|       | the            |
|       | organisation   |
|       | won            |
|       | the            |
|       | 2001           |
|       | nobel          |
|       | peace          |
|       | prize          |
|       | and            |
|       | a              |
|       | number         |
|       | of             |
|       | its            |
|       | officers       |
|       | and            |
|       | agencies       |
|       | have           |
|       | also           |
|       | been           |
|       | awarded        |
|       | the            |
|       | prize.         |
|       | other          |
|       | evaluations    |
|       | of             |
```

aa	tuples1: tuple
	(the, 0)
	(organisation, 0)
	(won, 0)
	(the, 0)
	(2001, 0)
	(nobel, 0)
	(peace, 0)
	(prize, 0)
	(and, 0)
	(a, 0)
	(number, 0)
	(of, 0)
	(its, 0)
	(officers, 0)
	(and, 0)
	(agencies, 0)
	(have, 0)
	(also, 0)
	(been, 0)
	(awarded, 0)
	(the, 0)
	(prize., 0)
	(other, 0)
	(evaluations, 0)
	(of, 0)
	(the, 0)
	(un's, 0)
	(effectiveness, 0)
	(have, 0)
	(been, 0)
	(mixed., 0)

fin1	tuples: tuple
	(., 1)
	(193., 2)
	(1945, 3)
	(1945;, 1)
	(1947., 1)
	(1960s, 1)
	(1970s, 1)
	(1994, 1)
	(2001, 1)
	(2007., 1)
	(24, 3)
	(51, 2)
	(;, 5)
	(a, 5)
	(actions, 1)
	(after, 1)
	(agencies, 3)
	(agency, 1)
	(aid, 2)
	(allies, 1)
	(also, 1)
	(an, 4)
	(skipped, 1)
	(another, 2)
	(approving, 1)
	(april\ufffdjune, 1)
	(are, 2)
	(armed, 2)
	(as, 2)
	(assembly, 2)
	(assessed, 2)
	(at, 3)

final	word: chararray	count: int
	.	1
	193.	1
	1945	2
	1945;	1
	1947.	1
	1960s	1
	1970s	1
	1994	1
	2001	1
	2007.	1
	24	2
	51	1
	;	5
	a	4
	actions	1
	after	1
	agencies	3
	agency	1
	aid	1
	allies	1
	also	1
	an	3
	another	1
	approving	1
	april\ufffdjune	1
	are	1
	armed	1
	as	2
	assembly	2
	assessed	1
	at	2
	awarded	1
	ban	1
	bank	1
	be	2