Assignment 4 - CS 6240

- Sriharsha Srinivasa Karthik Kaipa, Sec 01

Running the code:

The source code for this assignment is available in the folder named "`assignment4`". The "`src`" folder contains the final code. Make sure the inputs are in the folder "`input`" alongside the "`src`" folder. I have set up "`spark-submit`" to be available directly from the command line. Run instructions for the code are as follows:

- For cleaning the project, building the jar and then running it from the command line, please use the following rules:
  ```
  :~$ make pagerank
  ```
- For cleaning the project, use the following rule
  ```
  make clean
  ```
- For building the jar, use the following rule
  ```
  make jar
  ```

Design Discussion:

`.textFile()` : This function commands Spark to read the input specified as arguments as a text file after applying any decompression if necessary and save each line in the input as an entry in an RDD.
`.map()` : This function applies the lambda function specified as an argument to every entry in the RDD that this function is called on. Since the data is distributed, the Spark framework applies this function individually on all the data in separate machines. The data is changed as specified by the lambda function. The lambda function has to be such that any change to each record of data should be independent so that there is no hit to optimization.
`.filter()` : This function is applied the same way as map() but acts as a filter to parse out unwanted records - in this case null values. It is also distributed and independent of the data in other machines.
`.persist()` : This function indicates to Spark that any data currently in any machine should be stored and not discarded so that it can be used for later computations. This is aimed at improving efficiency by not having to emit and receive repetitive data over the course of execution.
`.count()` : This function counts all the number of records in the entire RDD, spread across all the machines. Every machine sends out it's local count and they are all aggregated by the master machine.
`.mapValues()` : This function is similar to the map() function but it only modifies the values associated with the keys and the keys remain the same. It is also distributed and independent of the data in other machines.
`.doubleAccumulator()` : This function initializes a global accumulator, like global counters in Hadoop MapReduce. This is maintained by the master machine and is a thread safe counter.

`.join()` : This function is an RDD function which takes in another RDD and combines entries with the same key. Spark distributes the data from one RDD to machines such that the data stored in those machines doesn't have to be replicated over the network again.
`.flatMap()` : This function is similar to the map() function but it can take any number of values as input for the lambda function to work on, unlike map(). It is also distributed and independent of the data in other machines.
`.reduceByKey()` : This function is similar to the Reducer from Hadoop MapReduce. It acts on all the data in the associated RDD across all machines and executes the lambda function given as argument on all values associated with any key in the RDD. Unlike map(), this function acts across all machines because the specific key could be distributed across many machines.
`.subtractByKey()` : This function is similar to set difference in Set Theory, except that it takes the keys as elements to do the difference. It returns only those keys that are not in the RDD given as argument. This function has to act on all data across all machines.
`.union()` : This function is similar to set union in Set Theory, except that it takes keys as elements to do the union. It returns all unique keys in the RDD it is associated with and the RDD given as argument. Unlike map(), this function acts across all machines because the data is distributed across all machines.
`.sortBy()` : This function sorts all the data in the RDD it is associated with based on the keys by default or any other measure as specified by the lambda function given as argument. This sorting occurs on all the data spread across all machines.
`.take()` : This function polls only the first 'k' (argument) records from the RDD and returns them as an array. This function acts only on one machine and accesses others only and only if the number of records from the first machine fall short of the specified amount.
`.parallelize()` : This function converts an array into an RDD for doing any operations we want. This function takes as argument the array and an optional number of slices argument which can be used to make sure only one machine has all the entries in the array.
`.saveAsTextFile()` : This function saves all the data in the associated RDD into a text file. If the data is spread across multiple machines, it saves the data in separate files for each machine.

Comparing Hadoop and Spark:

```
map(line => parser(line)) - ReaderMapper map function in Hadoop MR

map { line =>
 if (line != null) {
   val parts = line.split(":")
   val len = parts(1).length - 1
   (parts(0).trim, parts(1).substring(1, len))
 } else null
}.filter(data => data != null) - ReaderMapper map function in Hadoop MR

Both the above were performed by the same function in Hadoop MR.

.persist() - This is missing from Hadoop MR
```

```
val total = links.count() - Global counters in Hadoop MR

var ranks = links.mapValues(v => 1.0 / total) - ReaderMapper map
function in Hadoop MR

val dangling = spark.doubleAccumulator("Dangling Node") - Achieved
using global counters in Hadoop MR

dangling.reset() - Taken care of by Hadoop as it resets global
variables every job run

val contribs = links.join(ranks).values.flatMap{ case (urls, rank) =>
 val branches = urls.split(",")
 val size = branches.length
 if (branches(0) != "")
   branches.map(url => (url.trim, rank / size))
 else {
   dangling.add(rank)
   None
 }.filter(data => data != null) - PageRankMapper map function in Hadoop
MR

.reduceByKey((a, b) => a + b)
.mapValues(rank => ((1 - randomness) / total) + (randomness * rank) +
((randomness * dangling.value) / total)) - PageRankReducer reduce
function in Hadoop MR for the most part. The dangling node calculation
in done in PageRankMapper map function for all runs as well as the
TopKMapper map function for the final run

ranks = ranks.subtractByKey(contribs)
 .mapValues(v => ((1 - randomness) / total) + ((randomness *
dangling.value) / total))
 .union(contribs) - PageRankReducer reduce function in Hadoop MR for
the most part. The dangling node calculation in done in PageRankMapper
map function for all runs as well as the TopKMapper map function for
the final run

val topk = ranks.sortBy(value => sortOrder * value._2).take(k) - A
combination of TopKMapper map function and TopKReducer reduce function
in Hadoop MR
```

Advantages and Disadvantages:

- In terms of ease of writing code and how short the code can be, Spark implementation is much better than Hadoop since in Hadoop, different classes need to be created for mappers and reducers while in Spark, the driver program takes care of everything for us as if it were a sequential program.
- In terms of optimization, Spark is much better than Hadoop since in Spark, we can persist data that doesn't change in their respective machines so that the data can be

used for later calculations whereas this flexibility is missing from Hadoop implementation.

- The source code for Spark implementation is much easier to read in comparison to Hadoop code for anyone who is familiar with both Scala and Java since the function names used in Spark are representative up to a certain extent of what they are meant to do while in Hadoop, it all depends on how the programmer has chosen the names.

Performance Comparison:

1 master 10 workers:

```
Spark : 3314 seconds
Hadoop : 3130 seconds
```

1 master 5 workers:

```
Spark : 7036 seconds
Hadoop : 4136 seconds
```

Explanation:

The expected time comparison between the two versions of PageRank is that Spark runs much much faster than Hadoop version because of all the optimizations that Spark offers. But in my implementation of the PageRank algorithm, it seems that Spark version is taking much longer than Hadoop version. This could be happening because of inefficient implementation of the algorithm because of lack of experience and knowledge of Scala and how it works with multi machine execution.

Sample Wiki PageRank for Spark:

```
(United_States_09d4,0.0036771679479096165)
(Wikimedia_Commons_7b57,0.002923846673991145)
(Country,0.0024085079830847267)
(England,0.0016182471020084078)
(Europe,0.001609435474918203)
(United_Kingdom_5ad7,0.0015952026038704656)
(Water,0.0015858608678806074)
(Germany,0.0015798035824436294)
(France,0.0015441843888125586)
(Earth,0.0015117599843545338)
(Animal,0.0015093317577722699)
(City,0.0013920909939771303)
(Week,0.0012774050675116383)
(Asia,0.0011964124258005933)
(Sunday,0.001172400222706796)
(Monday,0.0011539402403438464)
(Wednesday,0.0011430178255799428)
(Wiktionary,0.0011417075075867056)
(Money,0.0011266517264649095)
(Friday,0.0011152989635736532)
```

```
(Plant,0.001115175168707719)
(Saturday,0.0011026081332547036)
(Thursday,0.0010882332434087488)
(Tuesday,0.0010804818844186528)
(Computer,0.0010769974216858166)
(English_language,0.0010754022936500494)
(Italy,0.0010568419592050896)
(India,0.0010422943943413454)
(Government,0.0010235383592043991)
(Number,9.92255796302903E-4)
(Spain,9.485773001742598E-4)
(Day,9.345285525200596E-4)
(Japan,9.227437242781936E-4)
(People,8.876473663235139E-4)
(Canada,8.783985115594358E-4)
(Human,8.763765165149537E-4)
(Wikimedia_Foundation_83d9,8.501137480862884E-4)
(China,8.370944037567439E-4)
(Energy,8.359502807930593E-4)
(index,8.357740580052039E-4)
(Australia,8.17661138822344E-4)
(Sun,8.101287007867329E-4)
(Food,8.079776915292901E-4)
(Science,8.000077550732057E-4)
(Mathematics,7.886574619756436E-4)
(Television,7.416712072580092E-4)
(Russia,7.277091239969552E-4)
(Year,7.026236632489316E-4)
(State,6.981596422171706E-4)
(Music,6.975513601638004E-4)
(Language,6.848966200138942E-4)
(Greece,6.846691010237086E-4)
(Capital_(city),6.839657854884649E-4)
(Scotland,6.758007062372272E-4)
(Metal,6.673951580606376E-4)
(Wikipedia,6.637835419736776E-4)
(Greek_language,6.563546166715127E-4)
(Planet,6.510444756503773E-4)
(2004,6.488987968158425E-4)
(Sound,6.32641651415739E-4)
(Religion,6.289564766389454E-4)
(London,6.250927339283389E-4)
(Africa,6.231837092205632E-4)
(Poland,5.90727457984319E-4)
(Geography,5.879547969374432E-4)
(Liquid,5.825878960059619E-4)
```

```
(20th_century,5.820178508757838E-4)
(Law,5.804288374188917E-4)
(World,5.71897858003582E-4)
(19th_century,5.678115939192707E-4)
(Scientist,5.641399543369485E-4)
(Society,5.639955531042872E-4)
(Atom,5.508677623665808E-4)
(History,5.429729740809681E-4)
(Latin,5.404227783532277E-4)
(Sweden,5.381071307441909E-4)
(Light,5.380375957064058E-4)
(War,5.300004898040406E-4)
(Culture,5.264997018569812E-4)
(Netherlands,5.262991913643036E-4)
(Turkey,5.12090802039762E-4)
(God,5.099519025677905E-4)
(Building,5.090293272021376E-4)
(Plural,5.060683947403012E-4)
(Information,5.016878343796489E-4)
(Chemical_element,4.91601873026144E-4)
(Portugal,4.902634314533772E-4)
(Centuries,4.888823940060701E-4)
(Inhabitant,4.8563317938802355E-4)
(Denmark,4.812057975689368E-4)
(Austria,4.7757541421762737E-4)
(Cyprus,4.7606456535933904E-4)
(Ocean,4.689276503124125E-4)
(Species,4.633308592794775E-4)
(Moon,4.6292427823186086E-4)
(Disease,4.620791053740889E-4)
(Biology,4.61452136703037E-4)
(Book,4.6135869315091425E-4)
(University,4.5950883365132627E-4)
(Capital_city,4.5757996942398033E-4)
```

Sample Wiki PageRank for Hadoop MR:

```
0.010656700141346977    United_States_09d4
0.008220639776057527    Wikimedia_Commons_7b57
0.0064952402353202566    Country
0.004711283065929041    England
0.0045635853871618965    Germany
0.004414657920218037    United_Kingdom_5ad7
0.004380909397817181    Europe
0.004359713264360057    France
0.004308947196780668    Water
0.004172974263868258    Animal
```

```
0.004124838763467    City
0.003990314502819405    Earth
0.0032022623206404612    Wiktionary
0.003190687587569432    Asia
0.0031397724967365728    Week
0.0030900864024562403    Money
0.0030672136089680358    Plant
0.0030177594605749116    Computer
0.003006232620767282    Sunday
0.002965622101995127    Monday
0.0029584070412357914    English_language
0.002938916857904719    Italy
0.0029376069177789255    Wednesday
0.002891313724770175    India
0.0028679582384330796    Friday
0.002859088683119939    Government
0.0028359297174876197    Saturday
0.002800200551367329    Thursday
0.0027797327121635543    Tuesday
0.0027158710035411282    Spain
0.002628420081539975    Number
0.002623675284060614    Japan
0.0025578671694649027    Canada
0.002441746192137675    People
0.002405821692748891    Human
0.0023446916812946664    Australia
0.002326782767279969    China
0.002308180786288308    Day
0.0023064364423891173    Wikimedia_Foundation_83d9
0.002241928372616844    Food
0.002174809526271151    Energy
0.0021670058929277227    Mathematics
0.0021543794871223734    Science
0.002136262375946121    Television
0.0021041390580040977    Sun
0.0020360651170750724    Capital_(city)
0.0020252283756537004    Music
0.0020069998433729277    Russia
0.0019582379703325797    State
0.0019081671466966736    Greece
0.0019024539949288237    index
0.0019005915011195112    Scotland
0.0018930928409946729    Year
0.001891613493292557    Language
0.0018364339229566622    Metal
0.0018346998701719212    2004
```

```
0.0017946652579968441    Wikipedia
0.0017803392350589503    London
0.0017796319188465941    Greek_language
0.0017453826941465605    Religion
0.0017393334979107248    Sound
0.0016949247662825155    Africa
0.0016665522626982648    Planet
0.0016394780227830278    20th_century
0.001634741537357645    Poland
0.0016095159378284234    19th_century
0.0016014789732554573    Law
0.0015636273637329805    Geography
0.0015584184765731562    Liquid
0.0015508915158835817    World
0.001530950799436273    Scientist
0.00150874870954171    Society
0.001492023174735682    Inhabitant
0.0014871899132532646    Latin
0.0014829147468236852    Netherlands
0.0014804876270619798    Sweden
0.0014783865575128813    War
0.0014709528105714936    History
0.0014363723901016283    Light
0.001436325981133582    Atom
0.0014225398458921927    Building
0.001417104455261456    Culture
0.0014159428457179684    God
0.0013843277505049129    Centuries
0.0013723761053671705    Information
0.001371173133459838    Capital_city
0.0013708669728049654    Turkey
0.0013551765855783241    Plural
0.00134247735034764    Portugal
0.0013353346242485215    Chemical_element
0.0013169168937867152    University
0.0013154875173614937    Denmark
0.0013129632257306593    Book
0.001312082444868015    Austria
0.0013042374491607416    Species
0.0012939934363504517    Disease
0.0012786131678996253    North_America_e7c4
0.0012645015761637707    Biology
0.0012585104653527607    Ocean
0.0012529807724905777    U.S._state_5a68
```

Full Wiki PageRank for Spark:

```
(United_States_09d4,0.0010837306550098857)
(2006,0.0010243522366366426)
(United_Kingdom_5ad7,5.619519369281815E-4)
(2005,4.767201536518274E-4)
(Biography,3.813814788604731E-4)
(France,3.6642125493092866E-4)
(England,3.5222357136808986E-4)
(Canada,3.493293899751968E-4)
(2004,3.312098777001547E-4)
(Unicode,3.260828969111918E-4)
(Germany,3.219017434546599E-4)
(Latin_alphabet,3.094088789796742E-4)
(International_Phonetic_Alphabet_96f8,2.854959164313859E-4)
(Australia,2.8433112838024114E-4)
(English_language,2.8209679888566754E-4)
(India,2.687576228853216E-4)
(2003,2.6150380794035373E-4)
(Japan,2.5405329451024027E-4)
(Wiktionary,2.3992281946608865E-4)
(Italy,2.2886742194065093E-4)
(Geographic_coordinate_system,2.205333522316161E-4)
(2002,2.1415493980779068E-4)
(Europe,2.1175212408406095E-4)
(Internet_Movie_Database_7ea7,2.1105567739201502E-4)
(2001,2.0941687278724474E-4)
(London,1.973552249270568E-4)
(World_War_II_d045,1.9725629552678136E-4)
(2000,1.8974501464477032E-4)
(Spain,1.8163120521905802E-4)
(Record_label,1.8080084659542753E-4)
(Russia,1.7877484702364933E-4)
(1999,1.775721150921137E-4)
(Wikimedia_Commons_7b57,1.7634684912822097E-4)
(K,1.6753884096235217E-4)
(Z,1.632951606688854E-4)
(T,1.6013088808728438E-4)
(I,1.5798726199657963E-4)
(C,1.5692919250478955E-4)
(M,1.5661798103412458E-4)
(S,1.5586411475263052E-4)
(Y,1.5469054785610835E-4)
(1998,1.5405076338155757E-4)
(F,1.5384760922757798E-4)
(Diacritic,1.5382057917046827E-4)
(A,1.5283574620087893E-4)
```

```
(G,1.5272962936454897E-4)
(E,1.5269128561795216E-4)
(Music_genre,1.5221339612985792E-4)
(H,1.514788424797075E-4)
(R,1.5124836281549793E-4)
(W,1.5089891518602424E-4)
(D,1.5034471181125356E-4)
(Q,1.4978499998121359E-4)
(L,1.4974066470943375E-4)
(V,1.4907024885598205E-4)
(B,1.4876961332632548E-4)
(N,1.4810504921041175E-4)
(J,1.4808253785065926E-4)
(1997,1.4726450137232618E-4)
(Roman_numerals,1.4719444260228096E-4)
(O,1.4688481502893785E-4)
(P,1.4661026276891994E-4)
(X,1.4561171135635354E-4)
(U,1.44507977921109E-4)
(Football_(soccer),1.4373747312246716E-4)
(Scotland,1.4207136881195747E-4)
(Television,1.4198446743635246E-4)
(Sweden,1.4065936146953563E-4)
(1996,1.3609331024838904E-4)
(Latin,1.3551138168167132E-4)
(New_York_City_1428,1.335500130049726E-4)
(French_language,1.3344332305851596E-4)
(China,1.3234623533334082E-4)
(Cyrillic_alphabet,1.3110739034390958E-4)
(1995,1.303533771060909E-4)
(Punctuation,1.2994655089990879E-4)
(Minuscule,1.2745043416258466E-4)
(Netherlands,1.268390391712091E-4)
(Palaeography,1.256807300338869E-4)
(Alphabets_derived_from_the_Latin_c33b,1.2408276488382183E-4)
(New_Zealand_2311,1.238938970671691E-4)
(1994,1.2362314220959702E-4)
(History_of_the_Latin_alphabet_ec67,1.2343633296221283E-4)
(Film,1.2118425166098483E-4)
(Unicode_Latin_88af,1.2079623855721425E-4)
(List_of_Latin_letters_d712,1.201722153649972E-4)
(1991,1.1977461439161358E-4)
(ISO_646_0cb4,1.1945593672293137E-4)
(Public_domain,1.1838886956983922E-4)
(Mathematics,1.1800579281949913E-4)
(1993,1.1733176312301403E-4)
```

```
(Poland,1.1586885075740465E-4)
(California,1.148771910402848E-4)
(1990,1.1463750179776228E-4)
(Scientific_classification,1.1403183868497124E-4)
(Norway,1.1304145700031875E-4)
(1992,1.1244483309498286E-4)
(German_language,1.1168471572424683E-4)
(Writing_system,1.1101449562244583E-4)
(Greek_language,1.1060347564060049E-4)
```

Full Wiki PageRank for Hadoop MR:

```
0.020328173945099632    United_States_09d4
0.01676857403516821    2006
0.009379956797138371    United_Kingdom_5ad7
0.007976707136631527    2005
0.007816284494847574    Biography
0.006403586799651899    Canada
0.006328172721341767    England
0.005875132531300701    France
0.005623137574084618    2004
0.005465648295730867    Geographic_coordinate_system
0.005193700055033483    Australia
0.005163551443629481    Germany
0.004632736551948406    India
0.004548057304136969    2003
0.00442459242921374    Japan
0.00387304648328624    Internet_Movie_Database_7ea7
0.0037118203429510376    2001
0.0036294043486943027    2002
0.0035209352107723737    Italy
0.0034763517281045696    Record_label
0.003460132535920062    2000
0.0034325887639080664    Europe
0.0032467589437966725    Population_density
0.003197766312305399    World_War_II_d045
0.0031443132743244552    London
0.003031902837474414    1999
0.0029645902411086045    Music_genre
0.00293111443324066    Spain
0.0027924565089871667    Race_(United_States_Census)_a07d
0.002767867474258461    English_language
0.0027088960240600193    Football_(soccer)
0.0026971219995995544    Russia
0.0026310676898812437    Wiktionary
0.002629982966837057    1998
0.002490265453607359    1997
```

```
0.002480644449033358    Scotland
0.00244392496105335    Television
0.002399543540459641    Census
0.0023872836566535315    New_York_City_1428
0.0023658860548404536    Sweden
0.0023461114805196906    Wikimedia_Commons_7b57
0.0023417826815513455    1996
0.00229899361301265    Square_mile
0.0022564758736645365    California
0.0021948342750785434    1995
0.002158589761611589    New_Zealand_2311
0.002133590863459327    Scientific_classification
0.002114859056100764    China
0.0020939112947327577    Actor
0.0020917731177998266    1994
0.002049047181290158    Netherlands
0.0020474821402696455    Public_domain
0.002041315387841829    Politician
0.0020032764602457664    Film
0.0020006233317782663    New_York_3da4
0.0019885924252686584    United_States_Census_Bureau_2c85
0.0019836226086137375    1993
0.0019644005879764937    1991
0.001947383042797961    1990
0.0019397407028829239    Norway
0.0019302329005678675    Album
0.0018940257913438161    1992
0.0018881123586562952    Marriage
0.0018878081770618937    Population
0.0018662397823301633    Ireland
0.0018657206747686049    Poland
0.0018608058759947799    Record_producer
0.0018358869318855484    Personal_name
0.0017978074335498295    Per_capita_income
0.0017889129420747539    Brazil
0.0017607738686730645    1989
0.0017199458511551401    Mexico
0.0017064644912262938    1980
0.0016905682817106625    Studio_album
0.001663916076440879    1986
0.0016612547348724238    Poverty_line
0.0016330279890918664    1985
0.001626331837988042    1982
0.0016174863282596252    1981
0.0016120809203950246    Animal
0.0016092398801357975    1979
```

```
0.0016079671463468603    South_Africa_1287
0.001603195334692069    1987
0.0016001705386681042    1983
0.0015993641704791716    1984
0.0015740804855251243    1974
0.00157222768911212    Switzerland
0.0015681051082448592    1988
0.001553505183337771    January_1
0.0015281069107824495    Latin
0.0015218300236394188    1970
0.0015158575159968366    1976
0.0014971983401575616    1975
0.0014851192435254832    Paris
0.0014725613132124326    French_language
0.0014718499399261108    1978
0.001470968169533231    Greece
0.0014685829069004885    1977
0.0014648182391194186    1972
0.0014613959487998417    1969
```

As can be seen, the pagerank values and the pages in the top 100 for Spark version and Hadoop version are slightly different. This may have happened because of some bug that crept into either of the two implementations.