Universität Hamburg
Department Informatik
Wissenschaftliches Rechnen
Dr. J. Kunkel, H. Hatzel, J. Gresens

Übungsblatt 5 zur Vorlesung
Big Data Analytics
im WiSe 2015/2016
Abgabe: 20.11.2015, 420 P

Bitte dokumentieren Sie die benötigte Bearbeitungszeit für die einzelnen Aufgaben in `bearbeitungszeit.txt`. Bitte denken Sie auch daran uns Feedback zur Veranstaltung zu geben:

`http://goo.gl/forms/B01IIDHvW2`

Bei der Abgabe von Source Code kommentieren Sie diesen bitte.

Dieser Übungszettel enthält eine Bonusaufgabe. Die hierbei erzielten Punkte können zu einem beliebigen Übungszettel addiert werden und somit fehlende Punkte ausgleichen.

1 IMDb Data Cleaning: `imdb[movie|actors].csv` (210 P)

In dieser Aufgabe wollen wir semi-strukturierte Daten über Filme aufbereiten. Diese werden im Folgenden noch nützlich sein.
Hierbei gehen wir wie folgt vor:

- Schreiben Sie ein objektorientiertes Python-Programm zum Einlesen der benötigten Dateien (diese sind in Tabelle 1 aufgeführt).
- Schreiben Sie eine Funktion, welche die einzelnen Daten für Filme korreliert und in die Datei `imdbmovie.csv` schreibt. Die Spalten der CSV Datei sollten folgendes Format aufweisen und Strings sollten korrekt escaped werden:

  ```
  1 Filmname,Bewertung,Anzahl Bewertungen,Jahr,Produktionsland,Genre,List SchauspielerID,List
      ↪ Schlüsselworte,List Zitate
  ```

  Die Listen sollten als JSON abgelegt werden. Texte sollten als UTF-8 abgespeichert werden.
- Schreiben Sie eine Funktion, welche die einzelnen Daten für Schauspieler korreliert und in die Datei `imdbactors.csv` in folgendem Format schreibt:

  ```
  1 SchauspielerID,Geburtsdatum,Geburtsort,Todestag,Liste mit Zitaten,Biographie (Text)
  ```

- Überlegen Sie sich einige Prüfungen um die Datenqualität zu verbessern. Implementieren Sie zwei sinnvolle Prüfungen und beschreiben Sie welche Daten korrigiert bzw. entfernt wurden.
- Messen Sie die Laufzeit ihres Programms und begründen Sie inwiefern für diese Aufgabe BigData-Technologie zum Einsatz kommen sollte. Denken Sie hierbei an die 5-Vs (Challenges für BigData).

1.1 Datensatz: IMDb

Dieser Datensatz enthält Daten der IMDb, das sind Informationen über Fernseh- und Kinofilme, Schauspieler und vergleichbares. Der Datensatz ist zu finden unter `/home/bigdata/IM`.
Die Informationen der IMDb sind hierbei unterteilt in verschiedene Dateien. Ein Auszug der verfügbaren Dateien ist in Tabelle 1 gegeben.
Jede Datei enthält einen Header, der kurz weitere Informationen zum Datensatz gibt. Typischerweise ist der Inhalt vergleichsweise selbsterklärend.

Datei	Inhalt
movies.list	Filmtitel bzw. Episode (Jahr)
actors.list	Schauspieler, Filmtitel (Gastauftritt oder Rolle)
actresses.list	Schauspielerin, Filmtitel (Gastauftritt oder Rolle)
biographies.list	Schauspielerinformationen – Abkürzungen zeigen Informationen zum Datenfeld an. NM: Name, QU: Zitate, DB: Geburtsdatum und Ort, DD: Todestag, BG: Biographie
countries.list	Filmtitel (Jahr), Land
genres.list	Filmname (Jahr), Genre
quotes.list	Film (Jahr), Liste mit Zitaten
ratings.list	AnzahlBewertungen, Rang, Filmtitel (Jahr)
keywords.list	Filmname (Jahr), Keyword

Tabelle 1: Ausschnitt aus den Dateien im Datensatz der IMDB

1.2 Hinweise

1.2.1 Codegerüst für Python

```python
#!/bin/env python3

import csv
import json

# We will use docstrings now for documenting objects
# those are used when calling help(object)
class MoviesFileObject():
    """
    Class for parsing a single movie list file
    """
    film = ""
    jahr = 1900

    def __init__(self, film, jahr):
        self.film = film
        self.jahr = jahr

    def parse(filename):
        """
        Return a list of movie objects parsed
        """
        with open("movies.list", "r", encoding="latin") as f
            pass

        return []

class CountriesFileObject():
    """
    Class for parsing a single country list file
    """
    def parse(filename):
        return []

# Classes that hold all information

class Movie():
    """
    Contain the final information after joinging all list files together.
    """
    film = "x"
    jahr = 1900
    produktionsland = "y,y"
    schauspielerIDs = []

    def toArray(self):
        """
        Convert all members to a list
        """
        return [self.film, self.jahr, self.produktionsland, json.dumps(self.schauspielerIDs)]

class IMDbParser():
    """
    Parser for selected files in the IMDb directory.
    It creates Movie and Actor objects
    """

    def __init__(self, directory):
        self.directory = directory

    def parse(self):
        """
        Parse the files in the directory and return a tuple (movie list, actor list)
        """
        return ([Movie()], [])

```

```
67
68 if __name__ == "__main__":
69     ip = IMDbParser("imdb")
70     (mlist, alist) =  ip.parse()
71
72     # Now write out CSV files using CSV writer
73     with open('imdbactors.csv', 'w') as csvfile:
74         w = csv.writer(csvfile, delimiter=',')
75         for a in alist:
76             w.writerow(a.toArray())
77
78     with open('imdbmovies.csv', 'w') as csvfile:
79         w = csv.writer(csvfile, delimiter=',')
80         for m in mlist:
81             w.writerow(m.toArray())
```

Abgabe:

`1-imdb-clean.py` Ihre Python Programm, welches die IMDb Daten in dem übergebenen Ordner verarbeitet und `imdb[movie|actors].csv` schreibt. Schreiben Sie die Antworten auf die Fragen bzw. Überlegungen als Kommentare.

2 MapReduce mit R und Python: Wikipedia-Koordinaten und NetCDF verknüpfen (210 P)

Anhand unserer bereits extrahierten Daten wollen wir zu Orten aus der Wikipedia gemittelte Wetterdaten extrahieren.
Dafür brauchen wir von Aufgabenblatt 4 die extrahierten Koordinaten aus der Wikipedia als CSV Datei. Nutzen Sie dementsprechend Ihre Ergebnisse vom letzten Aufgabenblatt.
Außerdem werden wir die Daten aus dem NetCDF Datensatz als csv benötigen. Sie finden die komprimierte Datei bereits im HDFS unter `/user/bigdata/`.
Nun gilt es ein neues CSV zu erstellen. Darin sollen sich zu jeder Wikipedia Koordinate, die wir extrahiert haben, die dazugehörigen Wetterdaten gespeichert sein.
Das CSV File soll das folgende Format haben:

```
1 Ort (Artikeltitel), Lon, Lat, Lon NetCDF, Lat NetCDF, Frühling five-number summary (Temp), ... Winter
      ↪ (Temp), Frühling five-number summary (Niederschlag), ..., Winter five-number summary
      ↪ (Niederschlag)
```

Mit five-number summary werden in der Statistik die Percentile: Min, Q1, Median, Q3, Max bezeichnet.
Sie können eine beliebige (sinnvolle) Methode verwenden um die Jahreszeiten zu unterscheiden.
Wählen Sie dabei jeweils den nächsten Messpunkt zu den im Artikel angegebenen Koordinaten. Der nächste im NetCDF vertretene Koordinatendinatenpunkt sollte in ihrem Mapper gefunden und dann mit dem Artikelnamen ausgegben werden.
Gehen Sie wie folgt vor:

- Schreiben Sie einen einfachen Mapper für Python um die Daten von *wiki-coordinates.csv* und *netcdf.csv* einzulesen und für die weitere Berechnung anzupassen.
- Implementieren Sie den Reducer in R um die Daten für einen Ort zusammen zu fassen.

2.1 Hinweise

Zum Starten Ihres Programms können Sie folgendes Snippet verwenden:

```
1 hadoop jar /home/kunkel/bigdata/jars/hadoop-streaming-2.7.1.2.3.2.0-2950.jar \
2 -mapper /home/username/mapper.py -reducer /home/username/reducer.R \
3 -input /user/bigdata/netcdf/netcdf.csv /user/username/wiki-coordinates.csv \
4 -output /user/<username>/weather-places
```

Für das Debugging verwenden Sie $yarn$ unter Angabe der Application ID oder

```
1 > yarn logs -applicationId <ID>
```

bzw. führen alles auf der Kommandozeile aus:

```
cat netcdf.csv wiki-coordinates.csv | ./map.py | sort | ./reduce.R
```

2.1.1 Codegerüst für R

```
#!/usr/bin/env Rscript

# Discard error messages for loading libraries (if needed)
sink(file=NULL, type="message")
library('stringi')
# Remove redirection
sink(type="message")

stdin=file('stdin', open='r')

# Processing one line at a time
while(length( lines=readLines(con=stdin, n=1) ) > 0){
  # Do analysis here

  # Output once you have the required data:
  cat(stri_paste(toString(summary(data)), sep="\t"), sep="\n")
}
```

2.2 Hinweise

Denken Sie darüber nach wie sichergestellt ist das eine Partition alle Werte zu einem Ort erhält und somit ein korrektes Mittel (bzw. eine Summary) gebildet werden kann.

Abgabe:

`2-map.py`	Ihr Mapper für die Wetter Informationen zu Orten.
`2-reduce.R`	Ihr Reducer für die Wetter Informationen zu Orten in R.

3 Bonusaufgabe Implementation eines JOINs in MapReduce (240 P)

Implementieren Sie einen Join um die Koordinaten der NetCDF Daten mit einer zweiten CSV-Datei, die (einige wenige) Einwohner mit deren Koordinaten enthält zu verknüpfen. Zusätzlich verknüpfen Sie die Koordinaten mit den Städten bzw. Länder-Informationen, aus `wiki-coordinates.csv`. Somit wird es möglich die Wetter-Bedingungen an den Standorten der Einwohner zu erhalten. Das erwartete Ausgabeformat ist:

```
PersonName,Geburtsort,Datum,Temperatur,Niederschlagsmenge
```

Nehmen Sie einige der Schauspieler aus Aufgabe 1 und finden Sie die durschnittlichen Wetterbedingungen an ihrem Geburtsort.
Welche Probleme treten beim Finden der Koordinaten auf?
Ausgabe `imdb-actor-weather.csv`

3.1 Hinweise

Eine gute Anleitung zur Implementation von JOINs finden Sie unter:
`http://codingjunkie.net/mapreduce-reduce-joins/`

Abgabe:

`3-join.Java`	Ihr MapReduce Programm welches als Eingabe die drei CSV-Dateien (Wetter, Einwohner und Wiki-Koordinaten) verarbeitet.