Practical session

Apache spark Installation:

- Let’s get started using Apache Spark,
- in just four easy steps…
- **http://spark.apache.org/docs/latest/**

Step 1:

- Install Java JDK 6/7 on MacOSX or Windows
- http://www.oracle.com/technetwork/java/javase/downloads/jdk10-downloads-4416644.html
- Follow the license agreement instructions
- Then click the download for your OS
- Need JDK instead of JRE (for Maven, etc.)
- This is much simpler on Linux…

```
sudo apt-get -y install openjdk-7-jdk
```

Step 2: Download Spark

- we’ll be using Spark 2.3.1
- See: http://spark.apache.org/downloads.html
- 1. download this URL with a browser
- 2. double click the archive file to open it
- 3. connect into the newly created directory

Step 3: Run Spark Shell

- We’ll run Spark’s interactive shell…
- ./bin/spark-shell
- then from the “scala>” REPL prompt,

Installation: Optional Downloads: Python

- For Python 2.7, check out Anaconda by Continuum Analytics for a full-featured platform:
- https://www.anaconda.com/download/
- Documentation: https://conda.io/docs/user-guide/install/windows.html
- Requirements: https://conda.io/docs/user-guide/install/index.html#system-requirements

Step 3: Run Spark Shell with python

- ./bin/pyspark

Installation: Optional Downloads: Maven

- Java builds later also require Maven,
- which you can download at: http://maven.apache.org/download.cgi

Loading a text file:

- Save these lines into a file called *triples.txt* in some folder, for example your home folder.

- AbrahamAbel born 1992
- AbrahamAbel livesIn duluth
- AbrahamAbel phone 789456123
- BetsyBertram born 1987
- BetsyBertram livesIn berlin
- BetsyBertram phone _anon001
- BetsyBertram phone _anon002
- _anon001 area 56
- _anon001 local 9874321
- _anon001 ext 123
- _anon002 area 56
- _anon002 local 1234789
- CharleneCharade born 1963
- CharleneCharade bornIn bristol
- CharleneCharade address _anon003
- CharleneCharade knows BetsyBertram
- _anon003 street brislingtonGrove
- _anon003 number 13
- _anon003 postCode bs4
- _anon003 postName bristol
- DirkDental born 1996
- DirkDental bornIn bergen
- DirkDental knows BetsyBertram
- DirkDental knows CharleneCharade

- In a console (or command prompt, or terminal) window, start the Spark shell:

-
  ```
  spark-shell
  ```
- From now on, we will run our commands inside the Spark shell, after the scala> prompt. Load the triples.txt file into Spark:

-
  ```
  val triples_str = sc.textFile("/home/sinoa/triples.txt")
  ```

- (You must use a forwards-slash: / even on Windows.)

- triples_str is now the name of a Resilient Distributed Dataset inside your spark-shell. You can enforce file loading and look at the resulting contents of the triples_str RDD with:

- triples_str.collect()

- (In Scala, you can always drop empty parentheses: (), which we will do from now - so triples.collect also works.)

Spark transformations and actions

- We are now ready to try out simple Spark transformations and actions: transformations create new RDDs when they are run, whereas actions produce side-effects or simpler variables.

- This action counts the number of lines in triples.txt (or strings in triples_str):

```
triples_str.count
```

- This transformation is likely to introduce duplicate lines:
- triples_str.sample(true, 0.9, scala.util.Random.nextInt).collect
- Save the result in a new RDD and rerun until truiples_dup contains at least one duplicate line:
- val triples_dup = triples_str.sample(true, 0.9, scala.util.Random.nextInt)
- triples_dup.collect
- This transformation removes duplicates:
- triples_dup.distinct.collect
- These are only a few of the simplest Spark transformations and actions. For a full list, see this tutorial page:
- https://www.tutorialspoint.com/apache_spark/apache_spark_core_programming.htm

- These actions get the first and 5 first lines in triples_str:

```
triples_str.first
triples_str.take(5)
```

- This action saves the triples into a subfolder of /home/sinoa/triples_copy:

```
triples_str.saveAsTextFile("/home/sinoa/triples_copy")
```

- This transformation creates a new RDD with a sample of non-duplicate lines from triples.txt.

```
triples_str.sample(false, 0.5, scala.util.Random.nextInt).collect
```

Unions and intersections

- Save these lines into a file called more_triples.txt:
- DirkDental born 1996
- DirkDental bornIn bergen
- DirkDental knows CharleneCharade
- EnyaEntity born 2002
- EnyaEntity address _anon001
- EnyaEntity knows CharleneCharade
- EnyaEntity knows DirkDental
- _anon001 street emmastrasse
- _anon001 number 7
- _anon001 postArea _anon002
- _anon002 postCode 45130
- _anon002 postName Essen

- These transformation produces all the lines that are in both files and all the lines that are in either file:

```
val more_triples = sc.textFile("/home/sinoa/more_triples.txt")
triples_str.union(more_triples).collect
triples_str.intersection(more_triples).collect
```

Tasks:

1) Use Spark's *flatMap* transformation to collect an array of distinct resources from the triples (i.e., those strings starting with a capital letter).
2) After reduction, the triple *<CharleneCharade knows BetsyBertram>* only appears for *CharleneCharade*. We want it to appear for *BetsyBertram* too.
3) After reduction, the triple *<_anon001 area 56>* appears for *_anon001*. We want to eliminate *_anon001* so that it appears for *BetsyBertram* instead. The trick is to use Spark's *join* transformation in the right way.
4) Include the triples from *more_triples.txt* in the map-reduce too. Note that *_anon001* occurs in both files, but represents a *different* anonymous node.
5) Make sure that your map-reduce job also eliminates *nested* anonymouse nodes: *more_triples.txt* has two levels of anonymouse nodes, so that the triple *<_anon002 postCode 555>* appears for *EnyaEntity*.