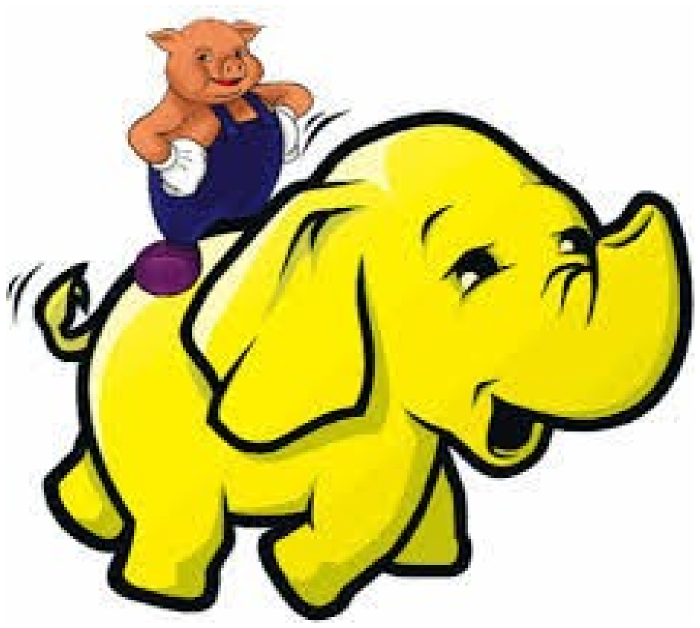
Carnegie Mellon University

95-885 Data Science and Big Data

Introduction to Pig



Objectives

Introduction to Pig

Structure of a Pig script

- Pig Data types
- Statements
- Examples
- Running a Pig script on Hadoop

Structure of a SQL statement

```
   SELECT   <columns> .................... 5
     FROM   <table>  ..................... 1
     JOIN   <table_2> ON <predicate> .... 1
    WHERE   <predicate on rows> ......... 2
 GROUP BY   <columns> .................... 3
   HAVING   <predicate on groups> ....... 4
 ORDER BY   <columns> .................... 6
```

In Pig each of these clauses becomes multiple map / reduce steps

Pig Data Types

- Scalar Types:
  - int, long, float, double, boolean, null, chararray, bytearray;
- Complex Types: fields, tuples, bags, map;
  - A Field is a piece of data
  - A Tuple is an ordered set of fields
  - A Bag is a collection of tuples
  - A Relation (or an alias) is a implemented as a bag
- Samples:
  - Tuple → Row in Database
    - (0002576169, John, 50, 3.8)
  - Bag → Table or View in Database

    {(0002576169 , John, 50, 3.8, Information Systems),

    (0002576170, Mike, 45, 3.6, Business, Freshman),

    (0002576171 Lucy, 36, 4.0), …. }

Variable schema
(Schema on read)

Pig Operations

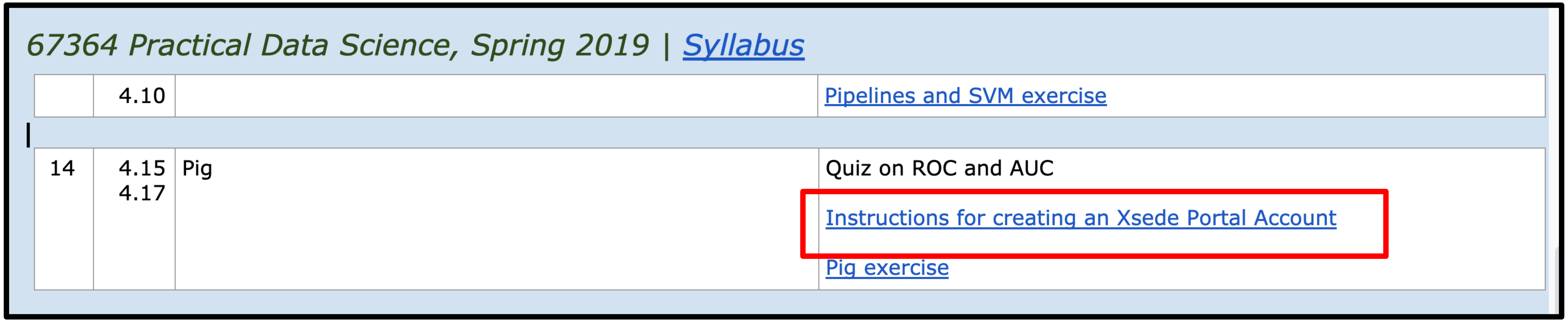
- Loading data
  - LOAD loads input data
  - Lines=LOAD ‘input/access.log’ AS (line: chararray);
- Projection
  - FOREACH … GENERATE … (similar to SELECT)
  - takes a set of expressions and applies them to every record (tuple).
- Grouping
  - GROUP collects together records with the same key
- Dump/Store
  - DUMP displays results to screen, STORE save results to file system
- Aggregation
  - AVG, COUNT, MAX, MIN, SUM

Word Count using Pig

```
Lines   = LOAD ‘$input’ AS (line: chararray);
Words   = FOREACH Lines GENERATE FLATTEN(TOKENIZE(line)) AS word;
Groups  = GROUP Words BY word;
Counts  = FOREACH Groups GENERATE  group, COUNT(Words) as cnt;
Results = ORDER Counts BY cnt DESC;
Top5    = LIMIT Results 5;
STORE Top5 INTO ‘$output’;
```

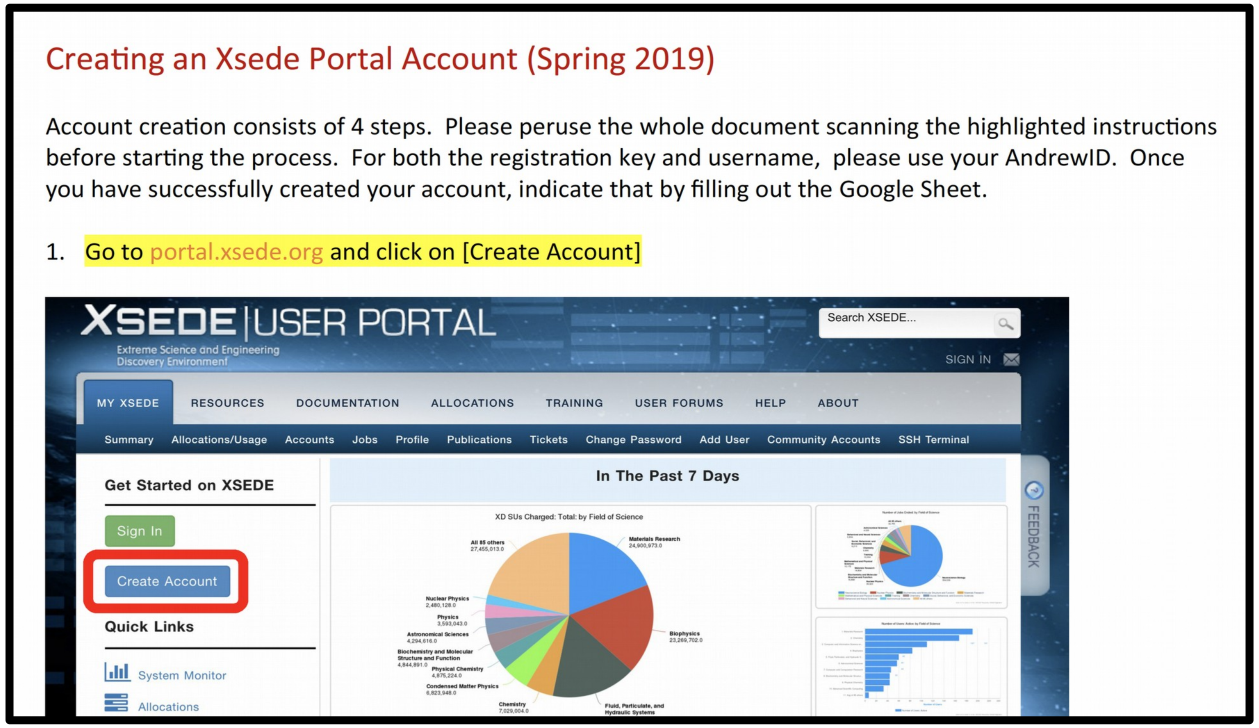
HADOOP ON BRIDGES

Instructions for Creating an Xsede Portal Account



67364 Practical Data Science, Spring 2019 | *Syllabus*

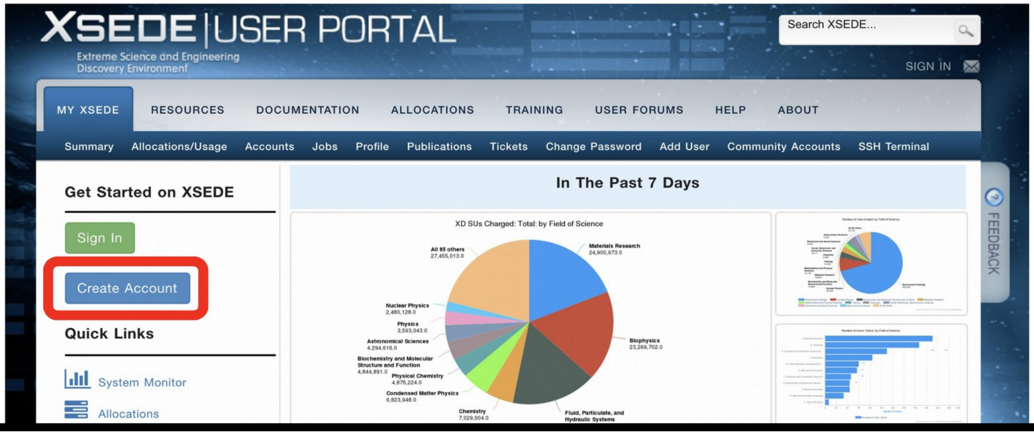
	4.10		Pipelines and SVM exercise
14	4.15 4.17	Pig	Quiz on ROC and AUC Instructions for creating an Xsede Portal Account Pig exercise



Creating an Xsede Portal Account (Spring 2019)

Account creation consists of 4 steps. Please peruse the whole document scanning the highlighted instructions before starting the process. For both the registration key and username, please use your AndrewID. Once you have successfully created your account, indicate that by filling out the Google Sheet.

1. Go to portal.xsede.org and click on [Create Account]



Logging on our Bridges' Hadoop cluster

1. Logon to bridges from your CLI (command line interface)

```
% ssh raja@bridges.psc.edu
raja@bridges.psc.edu's password:
[raja@login005 ~]$
```

The actual number of your login node may be different

BRIDGES
NEXT-GENERATION SUPERCOMPUTING FOR NEW COMMUNITIES AND BIG DATA

2. Logon to the hadoop cluster (from a Bridges machine)

```
[raja@login005 ~]$  ssh r383   # you won't be prompted for a password
```

3. You are now logged on to the "name node" of the cluster

```
[raja@r383 ~]$
```

Your prompt should be for r383

4. From the Hadoop cluster activate your Hadoop commands with

```
[raja@r383 ~]$ source ~raja/init-hadoop
```

You need to run the above command each time you login

- *1+19 node cluster*
- *Each node has 28 cores and 128 GB Ram*
- *Hadoop cluster:*
  - *19 x 28 = 532 cores*
  - *19 x 128 = 1432 GB*

Dissecting Word Count …

```
Lines   = LOAD ‘$input’ AS (line: chararray);
Words   = FOREACH Lines GENERATE FLATTEN(TOKENIZE(line)) AS word;
Groups  = GROUP Words BY word;
Counts  = FOREACH Groups GENERATE  group, COUNT(Words) as cnt;
Results = ORDER Counts BY cnt DESC;
Top5    = LIMIT Results 5;
STORE Top5 INTO ‘$output’;
```

Pig Operations

- Pig Data Loader
  - PigStorage: loads/stores relations using field-delimited text format

```
(John,18,4.0F)
(Mary,19,3.8F)
(Bill,20,3.9F)
```

```
students = load 'student.txt' using PigStorage('\t')
                      as (studentid: int, name:chararray,
                          age:int, gpa:double);
```

  - TextLoader: loads relations from a plain-text format
  - BinStorage:loads/stores relations from or to binary files
  - PigDump: stores relations by writing the toString() representation of tuples, one per line

LOAD

LOAD 'data' [USING function] [AS schema];

- data – name of the directory or file
  - Must be in single quotes
- USING – specifies the load function to use
  - By default uses PigStorage which parses each line into fields using a delimiter
  - default delimiter is tab (‘\t’)
  - The delimiter can be customized using regular expressions
- AS – assign a schema to incoming data
  - Assigns names to fields
  - Declares types to fields

Pig Operations - Foreach

- FOREACH ... GENERATE
  - iterates over the members of a bag

```
studentid = FOREACH  students  GENERATE
                              studentid, name;
```

  - The result of a Foreach is another bag
  - Elements are named as in the input bag

Pig Operations – Positional Reference

- Fields are referred to by positional notation or by name.

```
students = LOAD 'student.txt' USING PigStorage() AS (name:chararray, age:int, gpa:float);
DUMP A;
(John,18,4.0F)
(Mary,19,3.8F)
(Bill,20,3.9F)
studentname = Foreach students Generate $1 as studentname;
```

	First Field	Second Field	Third Field
Data Type	chararray	int	float
Position notation	$0	$1	$2
Name (variable)	name	age	Gpa
Field value	Tom	19	3.9

TOKENIZE & FLATTEN

- TOKENIZE returns a new bag for each input; “FLATTEN” eliminates bag nesting
- A:{line1, line2, line3…}
- After Tokenize:{{line1word1,line1word2,…}}, {line2word1,line2word2…}}
- After Flatten{line1word1,line1word2,line2word1…}

Pig Operations- Group

- Groups the data in one or more relations
  - The GROUP and COGROUP operators are identical.
  - Both operators work with one or more relations.
  - For readability GROUP is used in statements involving one relation
  - COGROUP is used in statements involving two or more relations. Jointly Group the tuples from A and B.

```
B = GROUP A BY age;
C = COGROUP A BY name, B BY name;
```

Pig Operations – Dump & Store

- DUMP Operator:
  - display output results, will always trigger execution
- STORE Operator:
  - Pig will parse entire script prior to writing for efficiency purposes

```
A = LOAD ‘input/pig/multiquery/A’;
B = FILTER A by $1 == “apple”;
C = FILTER A by $1 == “apple”;
SOTRE B INTO “output/b”
STORE C INTO “output/c”

Relations B&C both derived from A
        Prior this would create two MapReduce jobs
        Pig will now create one MapReduce job with output results
```

Pig Operations - Count

- Compute the number of elements in a bag
- Use the COUNT function to compute the number of elements in a bag.
- COUNT requires a preceding GROUP ALL statement for global counts and GROUP BY statement for group counts.

```
X = FOREACH B GENERATE COUNT(A);
```

Pig Operation - Order

- Sorts a relation based on one or more fields
- In Pig, relations are unordered. If you order relation A to produce relation X relations A and X still contain the same elements.

```
student = ORDER students BY gpa DESC;
```

How to run Pig Latin scripts

- Local mode
  - Local host and local file system is used
  - Neither Hadoop nor HDFS is required
  - Useful for prototyping and debugging
- MapReduce mode
  - Run on a Hadoop cluster and HDFS
- Batch mode - run a script directly
  - Pig –x local my_pig_script.pig
  - Pig –x mapreduce my_pig_script.pig
- Interactive mode use the Pig shell to run script
  - Grunt> Lines = LOAD ‘input.txt’ AS (line:chararray);
  - Grunt> Unique = DISTINCT Lines;
  - Grunt> DUMP Unique;