BIG DATA WORKSHOP

ST. DAVID SECONDARY SCHOOL
APRIL 29, 2024

Instructor: Prof. Wojciech Golab
wgolab@uwaterloo.ca

UNIVERSITY OF WATERLOO

URLS TO BOOKMARK

Slides:

https://manta.uwaterloo.ca/slides.pdf

Linux command prompt:

https://manta.uwaterloo.ca/shell

OUTLINE

- Introduction
- Linux shell
- Unstructured data
- Structured data
- Machine learning (time permitting)

INTRODUCTION

MEET YOUR INSTRUCTORS

Prof. Wojciech Golab

- Faculty member at the University of Waterloo, Department of Electrical and Computer Engineering since 2012.

Mr. Richards Peter

- Master of Applied Science (MASc) candidate at the University of Waterloo, Department of Electrical and Computer Engineering.

SPONSORS

Ministry of Research, Innovation and Science
(Early Researcher Awards Program)

University of Waterloo

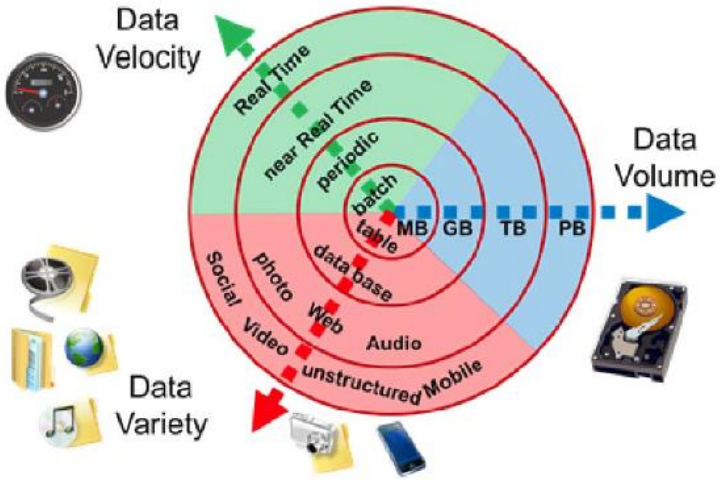
TYPES OF DATA

- Structured vs. unstructured.
  - » E-mail header has structure, but e-mail body has little structure.
  - » Spreadsheets and database tables are highly structured.
- Human-readable vs. binary.
  - » Social media posts are human-readable.
  - » Videos and audio files are binary.
- Human-generated vs. machine-generated.
  - » Books are human-generated.
  - » Satellite images are machine-generated.

THE THREE V'S OF BIG DATA

- Volume: amount of data generated.
- Variety: different types of data.
- Velocity: speed at which new data items are generated.

THE THREE V'S OF BIG DATA



Source: Wikipedia
https://en.wikipedia.org/wiki/Big_data

LINUX

WHAT IS LINUX?

- Open-source operating system kernel created by Linus Torvalds in 1991.
- Linux-based operating systems include the Linux kernel and powerful GNU tools.
- Extremely popular in servers, less so in desktop and laptop computers.
- Android uses the Linux kernel.

SETTING UP LINUX

Option 1: do it yourself.

- find an old computer and USB drive (min 4GB)
- back up old files, if any
- download a Linux ISO image to another computer
- copy the ISO image to the USB drive (https://rufus.ie/)
- boot from the USB drive and follow instructions

Option 2: run a Linux emulator inside a browser.

- https://bellard.org/jslinux/
- http://copy.sh/v86/

Option 3: rent from a cloud provider (e.g., Amazon).

- usually requires credit card
- provider may charge you for storage, network, and virtual machine rental

ACCESSING OUR LINUX SERVER

1. Open a web browser.

2. Navigate to https://manta.uwaterloo.ca

3. Enter your assigned user name at the login prompt, and then the password.

   Ctrl+V does not work, right click mouse instead

4. Create a second tab and repeat the process to create a second terminal.

THE LINUX SHELL

- The shell is one of several interfaces to the operating system (OS).
- The Linux shell provides a command prompt that executes programs based on your input.
- Bourne Again shell (bash) is one of the most popular Linux shells and will be used for exercises in this workshop.
- Multiple shell commands can be combined into programs called shell scripts.

COMMON SHELL COMMANDS

- ls – lists contents of current working directory
- pwd – print current working directory
- cd – change directory
- cat – concatenate files, or print file contents to the screen
- mv – move a file or directory
- cp – copy a file or directory
- rm – deletes a file or directory (permanently!)
- man – access manual page for a command (press *q* to quit and go back to the prompt)
- exit – terminate shell
- [Ctrl + c] – terminate a running program
- [up/down arrow keys] – access command history
- [tab key] – command completion

EXAMPLES

- pwd
- pwd --help
- man pwd
- ls /usr/bin
- ls -lh /var/log
- cat /etc/hostname
- who
- cp /data/zoo.csv .

INPUT AND OUTPUT REDIRECTION

- The < and > symbols are used to redirect the input and output streams of a program.
  - » man cat > cat_man_page.txt
  - » cat < cat_man_page.txt
- The | symbol pipes the output of one program to the input of another.
  - » ls -lh | tr a-z A-Z
  - » echo Loved | stem

EDITING FILES

- There are several popular Linux text editors including *nano*, *emacs*, and *vim*.
- These editors use keyboard shortcuts extensively, and do not require a mouse.
- Nano is the easiest for existing Windows users:
  - » arrow keys: position cursor
  - » Ctrl + arrow keys: skip over words and paragraphs
  - » Ctrl + s: save file
  - » Ctrl + x: exit editor
  - » Ctrl + k: cut line of text (copy to clipboard)
  - » Ctrl + u: uncut line of text (paste from clipboard)

SHELL SCRIPTS

- Use nano to edit myscript.sh in one terminal.
- In the other terminal, update permissions and run the script.
  - » chmod u+x myscript.sh
  - » ./myscript.sh

UNSTRUCTURED DATA

EXAMPLES OF (MOSTLY) UNSTRUCTURED DATA

- e-mails
- text messages
- chatroom conversations
- news articles
- books
- photos
- videos
- traffic and weather data

COMMON DATA OPERATIONS

- sampling
- translating
- sorting
- searching for keywords
- counting

SAMPLING DATA

- Print an entire file to the screen:
  - » cat myfile.txt
- Print the first 10 lines of a file:
  - » head myfile.txt
- Print the first 5 lines of a file:
  - » head -n 5 myfile.txt
- Print the first 5 lines of a file and capture output to another file:
  - » head -n 5 myfile.txt > myshorterfile.txt

SAMPLING DATA

- Print all but the last 5 lines of a file:
  - » head -n-5 myfile.txt
- Print the last 5 lines of a file:
  - » tail -n 5 myfile.txt
- Print the lines of a file in a random order:
  - » shuf myfile.txt
- Print 5 randomly sampled lines of a file:
  - » shuf -n 5 myfile.txt

TRANSLATING

- Convert text to upper case:
  - » cat myfile.txt | tr a-z A-Z
- Convert text to upper case (alternative):
  - » cat myfile.txt | tr '[:lower:]' '[:upper:]'
- Replace whitespace with line breaks:
  - » cat myfile.txt | tr '[:space:]' '\n'
- Replace whitespace with line breaks, squeeze repeated characters:
  - » cat myfile.txt | tr -s '[:space:]' '\n'

TRANSLATING

- Replace all non-alphanumeric characters with line breaks, squeeze repeated characters:
  - » cat myfile.txt | tr -s -c '[:alnum:]' '\n'

- First convert to uppercase, then replace all punctuation:
  - » cat myfile.txt | tr a-z A-Z | tr -d '[:punct:]'

SORTING

- Sort lines in lexicographical order:
  - » sort myfile.txt
- Sort lines in reverse lexicographical order:
  - » sort -r myfile.txt
- Sort integers in numerical order (i.e., 2 before 10):
  - » sort -n myfile.txt
- Sort lines and remove duplicates:
  - » sort myfile.txt | uniq

SEARCHING FOR KEYWORDS

- Find lines containing “Big”:
  - » grep Big myfile.txt
- Find lines containing “big data”, ignoring case:
  - » grep -i "big data" myfile.txt
- Find lines not containing “Data”:
  - » grep -v Data myfile.txt

SEARCHING FOR KEYWORDS

- Find lines starting with “Big”:
  - » grep "^Big" myfile.txt
- Find lines ending with “Workshop”:
  - » grep "Workshop$" myfile.txt
- Find lines containing “Big Data Workshop” and nothing else:
  - » grep "^Big Data Workshop$" myfile.txt

SEARCHING FOR KEYWORDS

- Find lines containing any three-letter word that starts with B and ends with g:
  - » grep "B[a-z]g" myfile.txt
- Find lines containing Big or Data:
  - » grep "Big\|Data" myfile.txt
- Find lines containing Big and Data:
  - » grep Big myfile.txt | grep Data

COUNTING

- Count the number of lines in a file:
  - » wc -l myfile.txt
- Count the number of words in a file:
  - » wc -w myfile.txt
- Count the number of lines matching a pattern:
  - » grep -c "Big" myfile.txt

EXERCISES

Please save your solutions in your home directory on the Linux server as exercise1.txt and exercise2.txt.

Successful completion of the exercises is required to earn credit for the workshop.

EXERCISE 1

Count the total number of occurrences of the word “time” in the book Alice in Wonderland (/data/Alice_in_Wonderland.txt).

Hint: get each occurrence of “time” on its own line, and then count the number of lines output.

EXERCISE 2

Design a sequence of commands that will count the frequencies of different words occurring in a text file.

Example:

For /data/small_file.txt, the output should indicate two occurrences of Big and Data, and one occurrence of every other word in the file.

Hint: Study the “man page” for the *sort* and *uniq* commands, and try to use *uniq* for counting.

STRUCTURED DATA

EXAMPLES OF STRUCTURED DATA

- accounting and payroll information
- class rosters
- grade files
- any data organized into tables with labelled columns

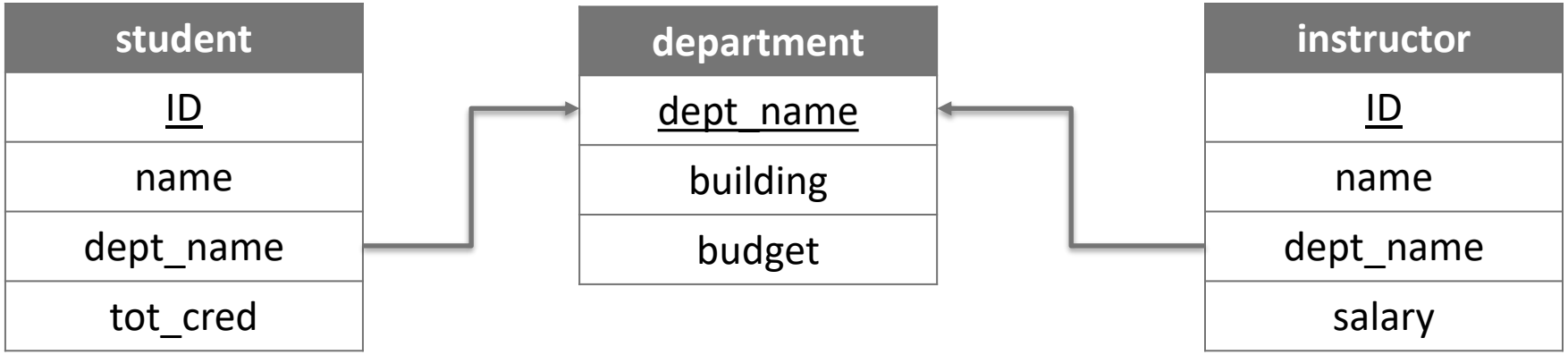
COMMON DATA OPERATIONS

- selection
- projection
- joins
- aggregation

SCHEMA: THE SHAPE OF DATA

- Structured data often has a well-defined *schema* that describes the organization of data.
- For example, databases organize data into tables, rows, and columns.
- Structured Query Language (SQL) is the most popular language for querying and manipulating structured data.
- MySQL is one of the most popular SQL databases.

THE UNIVERSITY SCHEMA
(FROM *DATABASE SYSTEMS CONCEPTS* BOOK BY SILBERSCHATZ, KORTH, AND SUDARSHAN, 2010)



The boxes in the diagram represent *tables*. The shaded header in each box is the table name and the items below the header are the names of *attributes* or columns in the table. The data types of the attributes are not shown, but are part of the schema. Underlined attribute are *primary keys*, which identify table rows uniquely. The arrows denote *foreign key constraints*, which indicate that the values of an attribute in one table (child table) must have counterparts among the values of an attribute in another table (parent table).

ACCESSING THE DATABASE

- In the Linux shell, launch the MySQL shell using following command:
  - » mysql

- When done, type “exit” or “quit” to terminate the MySQL shell and return to the Linux shell.

QUERYING THE SCHEMA

- Try running the following SQL commands inside the MySQL shell:
  - » describe student;
  - » describe instructor;
  - » describe department;

- Hint: terminate each statement with a semicolon!

THE ANATOMY OF AN SQL SELECT STATEMENT

select $attribute_1$, $attribute_2$, ..., $attribute_n$
from $table_1$, $table_2$, ..., $table_m$
where predicate ;

SELECTION

- Selection is an operator that filters out certain rows in a table.
- Filtering is achieved by adding a *where clause* to a query.

SELECTION

- Example: find complete student data for all students:
  - » select * from student;
- Example: find complete student data for computer science students:
  - » select * from student
    where dept_name = 'Comp. Sci.';
- Example: find complete student data for all students, sort by ID in descending order:
  - » select * from student order by ID desc;

PROJECTION

- Projection is an operator that filters out certain attributes in a table.
- Projection is achieved by naming the desired attributes immediately following the *select* keyword.

PROJECTION

- Example: find the IDs and names of all students:
  - » select ID,name from student;
- Example: find the names of all departments:
  - » select dept_name from department;
- Example: find the names of all departments that have at least one student:
  - » select dept_name from student;
- Example: find the names of departments that have at least one student, and eliminate duplicates:
  - » select distinct dept_name from student;

JOINS

- Join is an operator that combines data from multiple tables.
- Joins often match rows in two tables based on equality of certain attributes. The attributes compared often have the same name (e.g., dept_name in student and department).

JOINS

- Example: find the names of all students, their departments, and their department budgets.

  Solution 1:

  » select name, student.dept_name, budget
  from student, department
  where student.dept_name =
  department.dept_name;

  Solution 2:

  » select name, dept_name, budget
  from student natural join department;

JOINS

Solution 3:

» select name, student.dept_name, budget
from student join department
on student.dept_name = department.dept_name;

Solution 4:

» select name, dept_name, budget
from student join department
using (dept_name);

AGGREGATION

- Aggregation is used to combine data from different rows of a table.
- Aggregation is often is used to compute totals or subtotals.

AGGREGATION

- Example: find the total number of students.
  - » select count(*) from student;
- Example: find the number of students in each department that has at least one student.
  - » select dept_name, count(*)
    from student
    group by dept_name;
- Example: find the total number of credits earned by all students in the Biology department.
  - » select sum(tot_cred)
    from student
    where dept_name = 'Biology';

EXERCISES

Please save your solutions in your home directory on the Linux server as exercise3.txt and exercise4.txt.

Successful completion of the exercises is required to earn credit for the workshop.

EXERCISE 3

Find the IDs and names of students whose department has a budget of at least 80000.

Hint: join the student table with the department table on the dept_name column.

EXERCISE 4

For each building that has at least one department, find the number of departments in that building and their total budget.

Hint: use aggregation with a “group by” clause.

MACHINE LEARNING

DATA SET

Author: Richard Forsyth

Available at: https://github.com/renatopp/arff-datasets/blob/master/classification/zoo.arff

ATTRIBUTES

1. animal name:	Unique for each instance
2. hair	Boolean
3. feathers	Boolean
4. eggs	Boolean
5. milk	Boolean
6. airborne	Boolean
7. aquatic	Boolean
8. predator	Boolean
9. toothed	Boolean
10. backbone	Boolean
11. breathes	Boolean
12. venomous	Boolean
13. fins	Boolean
14. legs	Numeric (set of values: {0,2,4,5,6,8})
15. tail	Boolean
16. domestic	Boolean
17. catsize	Boolean
18. type	mammal, bird, reptile, fish, amphibian, insect, or invertebrate

WHITEBOARD

HOW TO QUERY USING SQL

- In the Linux shell, launch the MySQL shell using following command:
  - » mysql
- Then, select the zoo database as follows:
  - » use zoo;
- Run queries as needed:
  - » select distinct type from animals;
  - » select animal from animals where venomous;

HOW TO BUILD A DECISION TREE

- In the Linux shell, run Weka:
  - » java weka.classifiers.trees.J48
  - » java weka.classifiers.trees.J48 -C 0.25 -M 2 -t /data/zoo_train.arff -T /data/zoo_test.arff
  - » java weka.classifiers.trees.J48 -C 0.25 -M 2 -t /data/zoo_train.arff -T /data/zoo_test.arff -p 1

EXERCISES

Please save your solutions in your home directory on the Linux server as exercise5.txt and exercise6.txt.

EXERCISE 5

Find values for the -C and -M parameters that yield 100% accuracy on the training data sets.

Hint: try higher -C and lower -M than the defaults.

EXERCISE 6

Build the decision tree classifier with -C 0.25 and -M 2, then analyze the output and determine why so many reptiles are misclassified.

Hint: how many reptiles are in the training set?