

# Welcome!

## 261 - Machine Learning at Scale

Summer 2023

---



# **Week 1**

## **Live Session**

### **Overview**

1. Introductions
2. Housekeeping
  - o Things to know
  - o Action Items
  - o Expectations
3. Goals of the course
4. HW 1 preview
5. Week 1 material
  - o Bias-variance
  - o Word Count
  - o Sort command in Linux
6. Appendix: setup map-reduce cluster in the cloud

# Introductions



# Teaching Team

---

Name	Day Job	w261 Role	Time Zone	Email/Slack
Jimi Shanahan	deep learning   computer vision   edge-based computing   large scale data science consultant	Lecturer   Course Creator   Course Coordinator	PST (SF)	<a href="mailto:jimi@ischool.berkeley.edu">jimi@ischool.berkeley.edu</a>
Ramki Gummadi	CEO Argonaut AI	Lecturer	PST (SV)	<a href="mailto:rkgmd1729@gmail.com">rkgmd1729@gmail.com</a>
Vinicio De Sola	Senior Data Scientist Aspen Capital (W205 Lecturer)	Lecturer	PST+3 (NY)	<a href="mailto:vinicio.desola@ischool.berkeley.edu">vinicio.desola@ischool.berkeley.edu</a>
Luis Villarreal	Cloud Enterprise Architect, Accenture	Lecturer	PST+2 (Texas)	<a href="mailto:luis.villarreal@ischool.berkeley.edu">luis.villarreal@ischool.berkeley.edu</a>
Zach Gallante	W261 Alum	TA	EST (Bost)	
Toby Petty	W261 Alum	TA	PST+3 (NY)	
Samuel Gomes	W261 Alum	TA	PST	

## **Office Hours: weekly**

**(access details will announced via Slack/DigitalCampus just prior to the start of each OH)**



Name	OH period
Jimi Shanahan, and Samuel	Wednesday 3 PM PST
Toby	Thursday 5 PM PST
Vinicio De Sola	Sunday 10 AM PST
Ramki Gummadi	Sunday 3PM PST
Luis (on infrastructure)	Wednesday 6 PM PST for weeks 1 -3

# Housekeeping

---

# **Register now (if you have not done so already)!!**



Welcome to the Summer 2023 edition of W261,

Re: Slack registration [2 Actions Required]

We'll be using Slack for important course communications in addition to the Announcements function from Digital Campus. We an action item here:

**Action Item: register yourself on the course slack channels**

- Please register on the following Slack channels and add yourself using your slack ID (note: the channels will remain open for the next 10 days)
  - datasci-261-2023-summer-announcements
  - datasci-261-2023-summer-main
  - datasci-261-2023-summer-infrastructure

See you in class shortly,

Your W261 Instructional Team:

Jimi, Ramki, Vinicio

# Download the Canvas App: set notifications to *notify immediately*

This screenshot shows the Canvas mobile application interface. On the left is a vertical navigation bar with icons for Account, Dashboard, Courses, Calendar, Inbox, History, Placement Portal, and Help. The main area displays a user profile for Jimi Shanahan (He/Him) with initials JS. Below the profile are sections for Notifications, Profile, Files, Settings, Shared Content, QR for Mobile Login, and Global Announcements. A "Use High Contrast UI" toggle is at the bottom.

Step 1

The first screenshot shows the "Notifications" section of the Canvas mobile app. The "QR for Mobile Login" option is highlighted with a red circle. This is labeled "Step 2".

The second screenshot shows the "QR for Mobile Login" screen. It displays a QR code with the text "To log in to your Canvas account when you're on the go, scan this QR code from any Canvas mobile app." Below the QR code is a message stating "This code expires in 9 minutes." This is labeled "Step 2".

The third screenshot shows the "Notifications" settings screen. Under "Settings for" it says "Account". Below that is a table of course activities with notification options. For "Announcement", there is a "Notify immediately" button. Other rows include "Due Date", "Grading Policies", "Course Content", "Files", "Announcement Created By You", "Grading", and "Invitation". This is labeled "Step 3".

# Live session expectations



Come prepared: do the reading, watch the async, ask questions

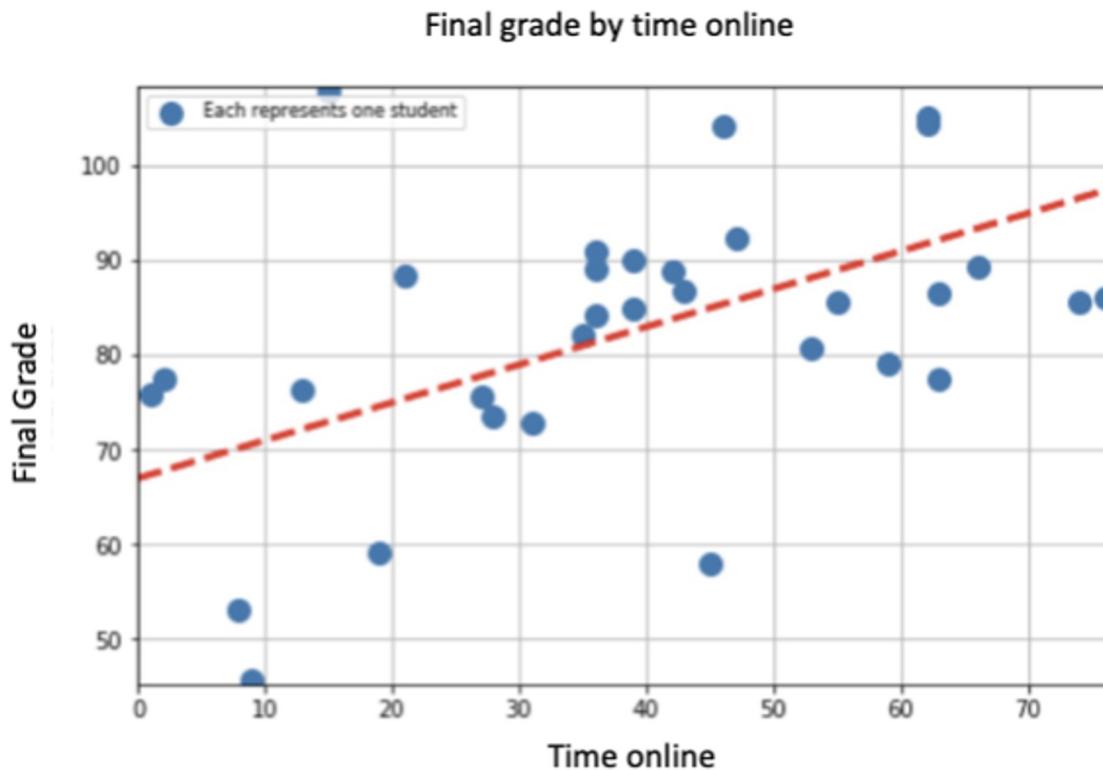
Be prepared for breakout sessions using in-class demo notebooks - have your environment and in-class demo notebooks up and running before class starts (you do not have to complete them).

Learn from each other: Ask and answer questions on Slack, but do not share code/answers

Come to office hours

Complete Homework on time

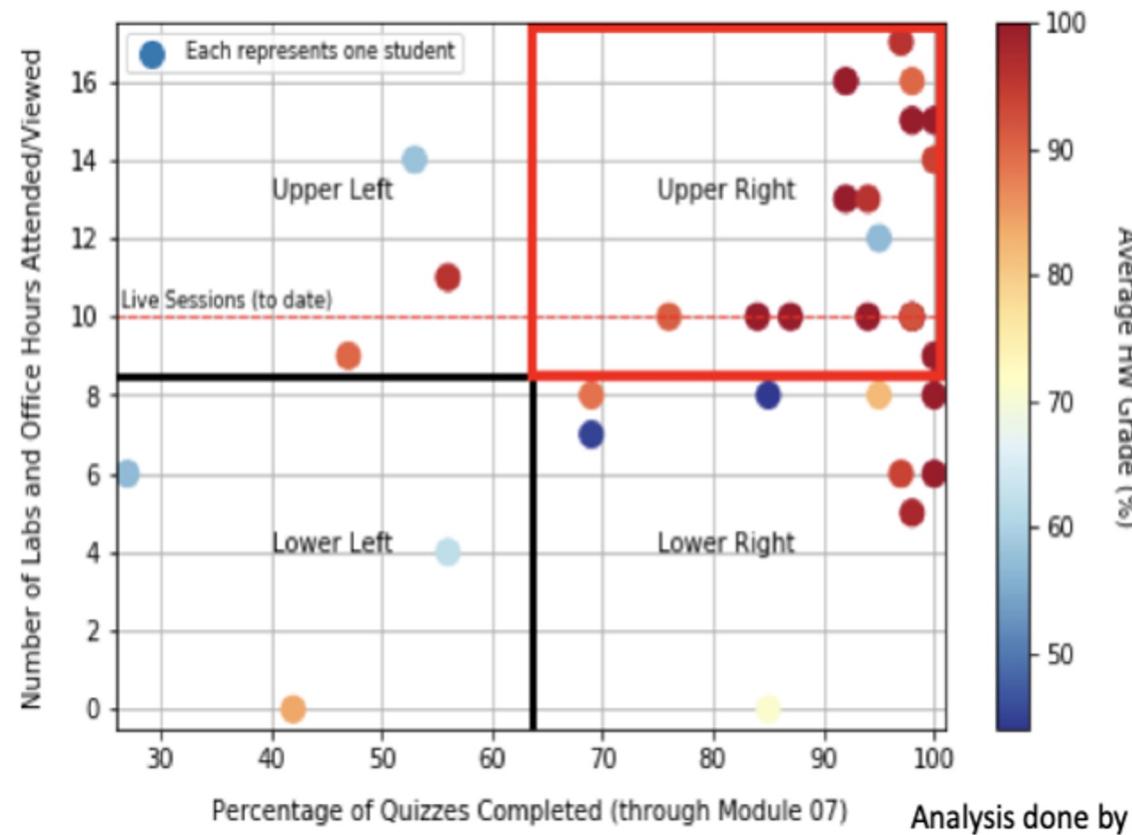
# Online masters students final grades analysis



Analysis done by James G. Shanahan



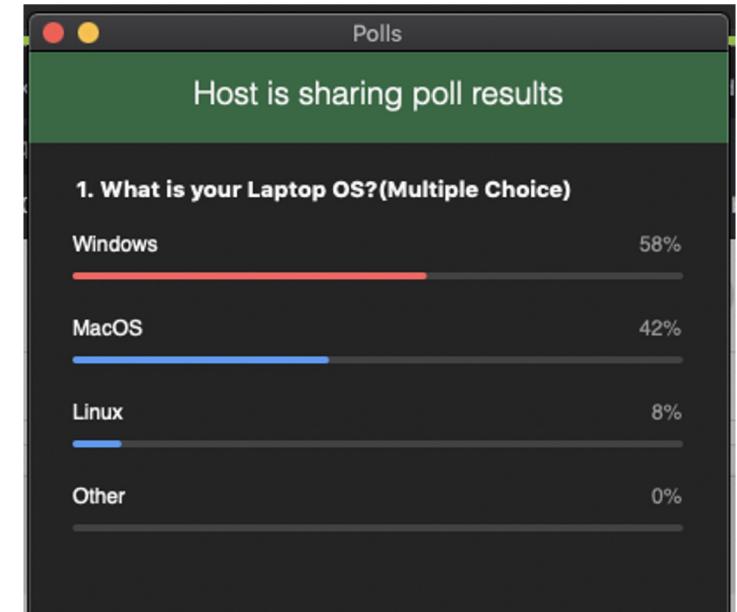
### Average Homework Grades by Lab and Quiz Participation



# Things to Know

---

- DigitalCampus (Canvas) is the source of all truth
- As for infrastructure, we will be using
  - Google Cloud (DataProc) until week 10
  - Google Colab (for Spark)
  - Databricks (running on Azure)
    - Access Instructions will be provided around week 10
    - Required the final project



# **Important Dates**

---

See Canvas

# Syllabus

- <https://digitalcampus.instructure.com/courses/4868/assignments/syllabus>

B

Account

Dashboard

Courses

Calendar

Inbox

History

Placement

≡ 2022-0822 DATASCI W261 Machine Learning at Scale > Syllabus

Student View Immersive Reader

Home Announcements Syllabus Modules Grades Files Zoom Live Sessions People Collaborations Faculty Course Community Pages

**Course Syllabus**

UC Berkeley School of Information | MIDS Program  
DATASCI w261  
Machine Learning at Scale  
Course Architect: Dr. James G. Shanahan, 1/1/2015  
Syllabus Last Revised on 8/22/2022 by Dr. James G. Shanahan.

Jump to Today Edit

<	31	1	2	3	4	5	6
	7	8	9	10	11	12	13
	14	15	16	17	18	19	20
	21	22	23	24	25	26	27
	28	29	30	31	1	2	3
	4	5	6	7	8	9	10

Assignments are weighted by group:

Group	Weight
Assignments	50%
Final Project	40%
Class Participation	10%

# Section 1: intro to ML, Map Reduce, Naïve Bayes

Module	Section	Async Lecture	Sync Lab + Review quiz	Assignments	Reading Materials
1	Intro and review	Machine Learning at Scale introduction and course overview	Command-line-based map-reduce, Google Cloud		<p>Read: <a href="#">ISL chapter 1 and sections 2.1 &amp; 2.2</a></p> <p>Skim: <a href="#">Adam Drake Blog Post</a> (<a href="#">Links to an external site.</a>)</p> <p>Optional: <a href="#">Fortmann-Roe Essay Clever Machine Bl</a></p> <p>Live Lab 1 SLIDES: See Module TAB</p>
2		Introduction to Hadoop Streaming	WordCount, secondary sorts, Naive Bayes	HW01 MapReduce via CMDLine due Midnight Sunday (of week 2)	<p>Read: <a href="#">DITP Chapter 1 &amp; Chapter 2</a></p> <p>Read: <a href="#">IIR CH.13</a></p> <p>Optional: <a href="#">Michael Noll Hadoop MR Tutorial</a></p>
3		Map-Reduce Algorithm Design patterns and map-shuffle-reduce	Relative frequencies, custom partitioning, order inversion, inverted index		<p>Read: <a href="#">DITP sections 2.4 - 2.7 and 3.1-3.4</a> (<a href="#">Links to an external site.</a>)</p> <p>Read: <a href="#">IIR sections 13.1 and 13.2</a></p> <p>Read: HDG - Part II Chapter 7 - How MapReduce</p> <p>Skim: <a href="#">Total Order Sort Guide</a>, <a href="#">EECS Map Reduce I</a></p> <p>OPTIONAL: <a href="http://blog.ditullio.fr/categories/mapreduce/">http://blog.ditullio.fr/categories/mapreduce/</a></p> <p>Slides: Lab Notebook is self-contained</p> 

# Section 2: Intro to Spark

Week 4: Spark					
4	Spark	Intro to Spark/Map-Reduce with RDDs (part 1)	Transformations, actions, RDDs, <u>dataframes</u> , datasets, broadcasting, closures	HW02 (Hadoop NB) <u>due Midnight Sunday.</u>	<p>Read: <a href="#">HP Spark chapter 2</a> Read: <a href="#">Spark RDD Programming Guide</a> Skim: <a href="#">Learning Spark ch 3 &amp; 4</a> Skim: <a href="#">DISCO Paper</a> Skim: <a href="#">DocSim Paper</a></p> <p>Additional Spark resources for weeks 4, and 5</p> <ul style="list-style-type: none"><li>• Holden Karau - Spark summit 2017 <a href="https://www.youtube.com/watch?v=4xsBQYdHgn8&amp;feature=youtu.be">[40 min]</a></li><li>• Debugging Spark Holden Karau - Dec 2017 <a href="https://www.youtube.com/watch?v=s5p15QT0Zj8&amp;list=WL&amp;index=7&amp;t=0s">[45 min]</a></li></ul>
5		Intro to Spark/Map-Reduce with RDDs (part 2)	KMeans Clustering, pairs and stripes		<ul style="list-style-type: none"><li>• Watch: <a href="#">Debugging Spark - Holden Karau - Dec 2017</a> [45 min]</li><li>• Skim Chapter 16 in <a href="#">IIR Book</a></li></ul>

# Live Sessions and their corresponding recordings

The screenshot shows a university calendar interface for August 2022. The left sidebar includes links for Account, Dashboard, Courses, Calendar, Inbox, History, Placement Portal, and Help. The main calendar view shows dates from 31 to 22. A pop-up window is displayed for the event on August 23:

w261 Section 1: 1 (W261.1)  
(W261.1) (W261.1) (W261.1)  
(W261.1)

Aug 23, 2pm - 3:30pm

Calendar 2022-0822 DATSCI W261 Machine Learning at Scale  
W261.1

Location ZOOM

Details Lets meet on Zoom:  
<https://ucberkeley-2u-com.zoom.us/j/914033277>

The video recording is located here:  
<https://ucberkeley-2u-com.zoom.us/rec/share/9TdeB0ZxjTB8Q6GHplQ2qoj77Ch>

Red circles highlight the Zoom link and the recording URL.



Today

← → September 2022

Week

Month

Agenda



Account



Dashboard



Courses



Calendar



Inbox



History

Placement  
Portal

10

<https://digitalcampus.instructure.com>

SUN	MON	TUE	WED	THU	FRI	SAT
28	29	30 <a href="#">2p MIDS w261 Sectio...</a> <a href="#">4p MIDS w261 Sectio...</a>	31 <a href="#">2p MIDS w261 OH W...</a>	1 <a href="#">6:30p Section 04 - Vi...</a>	2 <a href="#">2p Section 05 - Vinici...</a> <a href="#">4p Section 06 - Vinici...</a>	3
4 <a href="#">HW1 - Intro to the M...</a>	5	6 <a href="#">2p MIDS w261 Sectio...</a> <a href="#">4p MIDS w261 Sectio...</a>	7 <a href="#">2p MIDS w261 OH W...</a>	8 <a href="#">6:30p Section 04 - Vi...</a>	9 <a href="#">2p Section 05 - Vinici...</a> <a href="#">4p Section 06 - Vinici...</a>	10
11	12	13 <a href="#">2p MIDS w261 Sectio...</a> <a href="#">4p MIDS w261 Sectio...</a>	14 <a href="#">2p MIDS w261 OH W...</a>	15 <a href="#">6:30p Section 04 - Vi...</a>	16 <a href="#">2p Section 05 - Vinici...</a> <a href="#">4p Section 06 - Vinici...</a>	17
18 <a href="#">HW2 - Naive Bayes i...</a>	19	20 <a href="#">2p MIDS w261 Sectio...</a> <a href="#">4p MIDS w261 Sectio...</a>	21 <a href="#">2p MIDS w261 OH W...</a>	22 <a href="#">6:30p Section 04 - Vi...</a>	23 <a href="#">2p Section 05 - Vinici...</a> <a href="#">4p Section 06 - Vinici...</a>	24
25	26	27 <a href="#">2p MIDS w261 Sectio...</a>	28 <a href="#">2p MIDS w261 OH W...</a>	29 <a href="#">6:30p Section 04 - Vi...</a>	30 <a href="#">11a Teaching Online ...</a>	1

# HW Schedule



5 HWs

And a Final project with multiple phases

<https://www.ischool.berkeley.edu/intranet/students/mids/calendar>

# Modules

The screenshot shows a web browser window for the URL <https://digitalcampus.instructure.com/courses/4868>. The browser's address bar and various tabs are visible at the top. On the left, a dark blue sidebar menu lists various course management tools: Account, Dashboard, Courses, Calendar, Inbox, History, Placement Portal, and Help (with 10 notifications). The 'Courses' option is currently selected. The main content area is titled 'Home' and displays a list of course modules. Each module is represented by a card with a title, a checkmark icon, a plus sign, and a three-dot menu icon. The modules listed are:

- 1: Introduction and Motivation for Machine Learning at Scale
- 2: Parallel Computation Frameworks
- 3: Map-Reduce Algorithm Design
- 4: Introduction to Spark With RDDs: Part I
- 5: Introduction to Spark With RDDs: Part II
- 6: Distributed Supervised Machine Learning: Part I
- 7: Distributed Supervised Machine Learning: Part II

<https://digitalcampus.instructure.com/courses/4868>

# Mastery based grading

---

We follow an approach called mastery based grading which is a little different if you've never encountered it before. Basically for each question the score you get is not an indication of a "percent of the sub questions you got right" instead you should think of it as a categorical indicator. **Each question gets an overall score regardless of number of subquestions/parts.**

Each question has some one or two core concepts plus a lot of details. The goal is to avoid docking students multiple times for little details but still ensure that overall you get a clear message if you miss an important concept - so anything that shows confusion/error around the core learning concept gets a 50 while any other error or combination of errors gets a 90.

100 - illustrates understanding of core concepts, no errors

90 - illustrates understanding of core concepts, some errors in calculations, typos, etc

50 - core concepts missed

0 - not attempted

# Grades breakdown



% of Final Grade	Component
50%	Homework Assignments (5 assignments, 10% each)
40%	Final Project
10%	Live session attendance and participation

# Final Letter Grades breakdown for w261

Letter grades	Range	Interpretation
A+	<= 100%	to 98 % Excellent
A	< 98%	to 92.5% Excellent
A-	< 92.5%	to 90% Very Good
B+	< 90%	to 82.5% Good
B	< 82.5%	to 75% Good
B-	< 75%	to 67.5% Fair
C+	< 67.5%	to 57.5% Below Expectations
C	< 57.5%	to 50% Below Expectations
C-	< 50%	to 42.5% Below Expectations
D+	< 42.5%	to 32.5% Below Expectations
D	< 32.5%	to 25% Below Expectations
F	< 25%	to 0% Below Expectations
I	Incomplete	Work incomplete due to circumstances beyond the student's control, but of passing quality; not included in grade-point computation after fall 1973.

# Student Code of Conduct

<https://conduct.berkeley.edu/violations/>

Berkeley **Center for Student Conduct**

About Us   Campus Policies   FAQ   Forms   Request Services   Request Restorative Justice Resolution

## Student Code of Conduct Violations

Home / Student Code of Conduct Violations

### Non-Academic

The Berkeley Campus Code of Student Conduct provides [a list of non-academic code of student conduct policies](#).

### Academic

The following definitions were present in the 1989 Berkeley Campus Code of Student Conduct but are no longer present in the revised Code (September 2004). The following definitions are for information only.

Academic dishonesty is any action or attempted action that may result in creating an unfair academic advantage for oneself or an unfair academic advantage or disadvantage for any other member or members of the academic community.

*Below are types of academic dishonesty with examples of each. Please note that this list is not exhaustive.*

### Cheating

Cheating is defined as fraud, deceit, or dishonesty in an academic assignment, or using or attempting to use materials, or assisting others in using materials that are prohibited or inappropriate in the context of the academic assignment in question, such as:

- Copying or attempting to copy from others during an exam or on an assignment.
- Communicating answers with another person during an exam.
- Preprogramming a calculator to contain answers or other unauthorized information for exams.
- Using unauthorized materials, prepared answers, written notes, or concealed information during an exam.
- Allowing others to do an assignment or portion of an assignment for you, including the use of a commercial term-paper service.
- Submission of the same assignment for more than one course without prior approval of all the instructors involved.
- Collaborating on an exam or assignment with any other person without prior approval from the instructor.
- Taking an exam for another person or having someone take an exam for you.

### Plagiarism

Plagiarism is defined as use of intellectual material produced by another person without acknowledging its source, for example:

[Code of Student Conduct](#)

[SVSH Policy](#)

[Conduct Process](#)

[Restorative Justice Resolution Process](#)

[Report an Incident](#)

# Preview of HW 1

[https://github.com/UCB-w261/main/blob/master/Assignments/HW1/hw1\\_Workbook.ipynb](https://github.com/UCB-w261/main/blob/master/Assignments/HW1/hw1_Workbook.ipynb)



The screenshot shows a user interface for a learning management system. On the left is a dark blue sidebar with various icons and links:

- B
- Account
- Dashboard
- Courses
- Calendar
- Inbox
- History
- Placement Portal
- Help (with a red notification badge showing 10)

On the right is the main content area. At the top, there's a navigation bar with links like "ONCLogin", "Dashboard", "EXC...", "[X][Y] GroupVII", "GITA NorCal", "Home / Twitter", "Expression below...", "Untitled Diagram...", and "Multi-Class Recur...".

The main content area displays a course structure:

- 1: Introduction and Motivation for Machine Learning at Scale**
  - Weekly Introduction 1**
  - Big Data Definitions**
  - Sources of Big Data: Society**
  - Internet of Things**
  - Data Modeling Pipeline**
  - Data Scientist: Post This Class**
  - Goals of This Class**
  - Large-Scale Machine Learning**
  - Bias-Variance Background**
  - Question: Quiz 1**  
0 pts
  - Summary**
- HW1 - Intro to the Map Reduce Paradigm**  
Sep 4 | 100 pts

A red rectangle highlights the "HW1 - Intro to the Map Reduce Paradigm" assignment.

⋮	1: Introduction and Motivation for Machine Learning at Scale	✓	+	⋮
⋮	🔗 Weekly Introduction 1	✓	⋮	
⋮	🔗 Big Data Definitions	✓	⋮	
⋮	🔗 Sources of Big Data: Society	✓	⋮	
⋮	🔗 Internet of Things	✓	⋮	
⋮	Internet of Things 🔗 Data Modeling Pipeline	✓	⋮	
⋮	🔗 Data Scientist: Post This Class	✓	⋮	
⋮	🔗 Goals of This Class	✓	⋮	
⋮	🔗 Large-Scale Machine Learning	✓	⋮	
⋮	🔗 Bias-Variance Background	✓	⋮	
⋮	🔗 Question: Quiz 1 0 pts	✓	⋮	
⋮	🔗 Summary	✓	⋮	
⋮	🔗 HW1 - Intro to the Map Reduce Paradigm Sep 4, 2022   100 pts	✓	⋮	
⋮	📎 W261 - Machine Learning at Scale - Week 1 - Live Session.pdf	∅	⋮	
⋮	📎 Video and Slide: Cloud computing and JupyterLab PaaS on Google Cloud	✓	⋮	

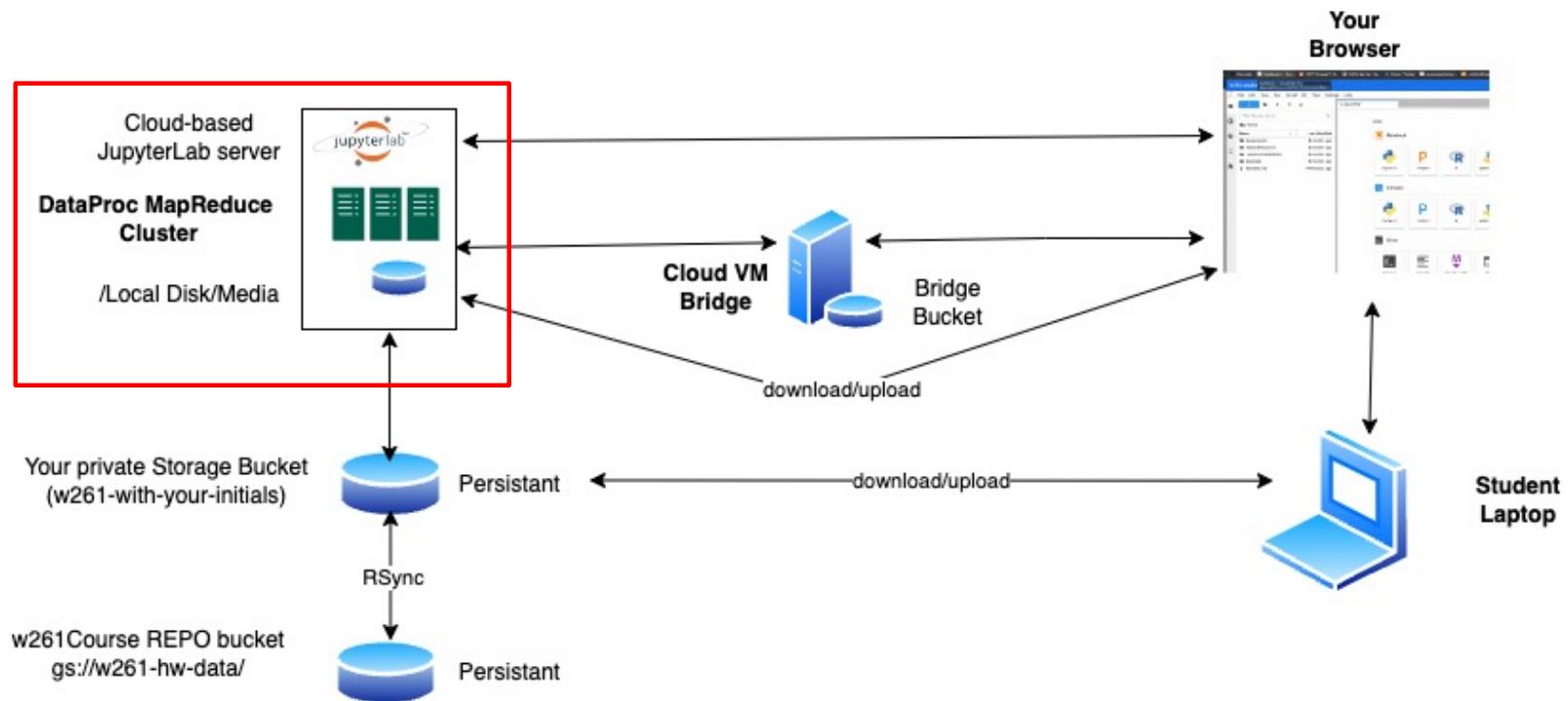
DataProc

# JupyterLab PaaS at \$0.20 per hour

**Dataproc** is a fully managed and highly scalable service for running Hadoop, Apache Spark, Apache Flink, Presto, and 30+ open source tools and frameworks.

Setup a hadoop/Spark (aka DataProc) cluster on Google cloud with a **single node** (4 CPUs)  
The cluster costs \$0.20 per hour and has 100 Gig of local Disk

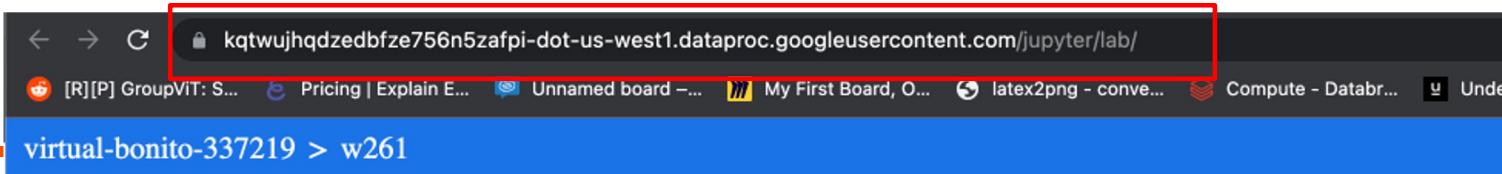
~\$35 per week  
~ \$300 for 9 weeks



Click [here](#) to access CloudShell/Bridge:  
<https://console.cloud.google.com/>

Orchestration shell script cmd:    `gsutil cat gs://w261-hw-data/w261_env.sh | bash -euo pipefail`

# Cloud-based JupyterLab (backed by a DataProc Cluster)



The screenshot shows a JupyterLab interface running on a cloud-based DataProc cluster. The URL in the browser is `kqtwujhqdzedbfze756n5zafpi-dot-us-west1.dataproc.googleusercontent.com/jupyter/lab/`. The interface includes a file browser on the left, a notebook tab titled "hw3\_Workbook.ipynb" in the center, and a terminal tab on the right with the command `bash -euo pipefail`.

A red box highlights the URL bar at the top of the browser window.

A pink box highlights the path `/Local Disk/media/notebooks/Assignments/HW1/` in the file browser sidebar.

A diagram at the bottom illustrates the setup:

- A blue arrow points from the "Cloud-based JupyterLab" icon to the "Student Laptop" icon.
- A red box encloses the "Cloud-based JupyterLab" icon and the "Student Laptop" icon.
- A red box highlights the text "... define the terms one-hot encoding, co-occurrence matrix".

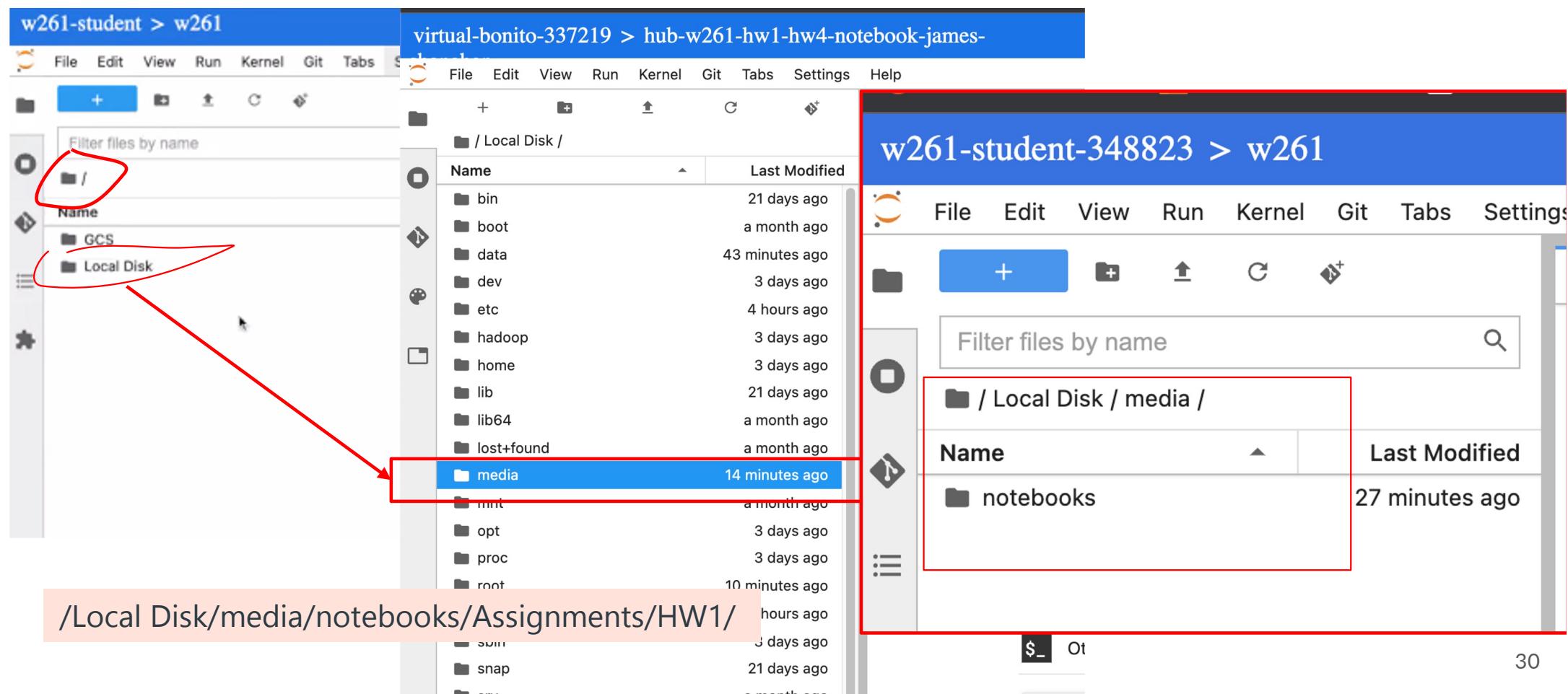
**HW 3 - Synonym Detection In Spark**

**MIDS w261: Machine Learning at Scale | UC Berkeley School of Information**

In the last homework assignment you performed Naive Bayes to classify documents. In this assignment you will learn how word embeddings work. A word embedding is a vector representation of a word such that words with similar meaning have similar vectors. In this assignment you will learn how to use word embeddings to detect synonyms. You will also learn how to use word embeddings to perform document similarity analysis. In doing so you'll also become familiar with other metrics for comparing vectors, such as cosine, Jaccard, and Dice. By the end of this homework you should be able to:

- ... define the terms one-hot encoding, co-occurrence matrix, word embeddings, and word2vec.

# Click on / folder and select Local Disk



# Directory structure for HW

---

```
W261-f21-jimi  # course-semester-mygitid
    --Assignments
        --HW1
            |__hw1_Workbook.ipynb ← change the name of the notebook to avoid collisions
            |__collateCounts.py
            |__pWordCount.sh
            |__wordCount.py
            |__README.md
            |__ <directory for scratchwork>
        --HW2
            |__ etc ...
    --HelpfulResources
    etc..
```

# HW submission via Canvas (LMS)

---

≡ w261 > Assignments

---

[Home](#)

[Announcements](#) ⏱

[Assignments](#)

[Discussions](#)

[Grades](#)

[People](#)

Search for Assignment

⋮ ▾ Assignments

⋮ ⏱ HW1 - Intro to the Map Reduce Paradigm

Due May 15 at 11:59pm | 98 pts

# HW1 notebook

## Question 1: Introductions

The Caterpillar and Alice looked at each other for some time in silence: at last the Caterpillar took the hookah out of its mouth, and addressed her in a languid, sleepy voice. "Who are you?" said the Caterpillar.

-- Lewis Carroll, *Alice's Adventures in Wonderland*, Chapter 4

Tell us about yourself! Briefly describe where you live, how far along you are in MIDS, what other classes you are taking and what you want to get out of w261.

Type your response here!

## Question 2: Bias - Variance

In 1-2 paragraphs (~200 and *absolutely no more than 300 words!*), explain the bias-variance trade off. Describe what it means to "decompose" sources of error. How is this used in machine learning? Please use mathematical equation(s) to support your explanation. (Use \$ signs to take advantage of *LATEX* formatting, eg. \$f(x)\$ will look like:  $f(x)$ ). Please also cite any sources that informed your answer.

Type your response here!

## Question 3: Tokenizing

A number of our assignments this term will involve extracting information from text. A common preprocessing step when working with raw files is to 'tokenize' (i.e. extract words from) the text. Within the field of Natural Language Processing a lot of thought goes into what specific tokenizing makes most sense for a given task. For example, you might choose to remove punctuation or to consider punctuation symbols 'tokens' in their own right. In this question you'll use the Python `re` module to create a tokenizer to use when you perform WordCount on the *Alice In Wonderland* text.

### Q3 Tasks:

- **a) short response:** In the Naive Bayes algorithm (which we'll implement next week), we'll estimate the *likelihood* of a word by counting the number of times it appears and dividing by the size of the vocabulary (total number of unique words). Using the text: "Alice had an adventure that took alice to wonderland", give a concrete example of how two different tokenizers could cause us to get two different results on this calculation. [ HINT : you should not need to read up on Naive Bayes to answer this question]
- **b) short response:** When tokenizing in this assignment we'll remove punctuation and discard numerical digits by making everything lowercase and then capturing only consecutive letters a to z. Suppose `tokenize(x)` is a Python function that performs the desired tokenization. What would `tokenize("By-the-bye, what became of Alice's 12 hats?!")` output? Type the answer in the space provided below.
- **c) code:** Fill in the regular expression pattern in the cell labeled `part c` so that the subsequent call to `re.findall(RE_PATTERN, ...)` returns the

### Question 1: Introductions

The Caterpillar and Alice looked at each other for some time in silence: at last the Caterpillar took the hookah out of its mouth, and addressed her in a languid, sleepy voice. "Who are you?" said the Caterpillar.

-- Lewis Carroll, Alice's Adventures in Wonderland, Chapter 4

Tell us about yourself! Briefly describe where you live, how far along you are in MIDS, what other classes you are taking and what you want to get out of w261.

### Question 2: Bias - Variance

In 1-2 paragraphs (~200 and *absolutely no more than 300 words!*), explain the bias-variance trade off. Describe what it means to "decompose" sources of error. How is this used in machine learning? Please use mathematical equation(s) to support your explanation. (Use  $\$$  signs to take advantage of  $L^A T_E X$  formatting, eg.  $\$f(x)\$$  will look like:  $f(x)$ ). Please also cite any sources that informed your answer.

### Question 3: Tokenizing Pick 3 questions, 5 pts per question

3.a



# Canvas form + Jupyter Notebook

## Question 26

1 pts

Please upload a copy of the Jupyter notebook in .HTML format that you used here.

Upload

File Edit View Run Kernel Tabs Settings Help

HW1\_WORKBOOK.IPYNB

1. HW1 - Intro to the Map Reduce Paradigm

2. Notebook Set-Up

3. Question 1: Introductions

4. Question 2: Bias - Variance

5. Question 3: Tokenizing

5.0.1. Q3 Student Answers:

6. Load the Data

7. Question 4: Word Count in Python

7.0.1. Q4 Student Answers:

8. Question 5: Unix Sorting

8.0.1. Q5 Student Answers:

9. Question 6: Parallel Word Count (part 1)

9.0.1. Q6 Student Answers:

10. Question 7: Parallel Word Count (part 2)

10.0.1. Q7 Student Answers:

11. Question 8: Scalability Considerations

11.0.1. Q8 Student Answers:

12. Question 9: Embarrassingly Parallel

12.0.1. Q9 Student Answers:

12.0.2. Congratulations, you have completed HW1! Please refer to the readme for

Labs\_S X PyTorch X Lab-pl X Unit14 X Phase X Phase X Boston X Linear X HW09 X hw1\_W X mappe X

Markdown C git

**1. HW1 - Intro to the Map Reduce Paradigm**

MIDS w261: Machine Learning at Scale | UC Berkeley School of Information | Spring 2018

Welcome to Machine Learning at Scale! This first homework assignment introduces one of the core strategies in distributed processing: divide and conquer. We'll use the simplest of tasks, word counting, to illustrate the difference between a scalable and non-scalable algorithm. You will be working with the text of *Alice in Wonderland* to put these ideas into practice using Python and Bash scripting. By the end of this week you should be able to:

- ... describe the Bias-Variance tradeoff as it applies to Machine Learning.
- ... explain why we consider word counting to be an "Embarrassingly Parallel" task.
- ... estimate the runtime of embarrassingly parallel tasks using "back of the envelope" calculations.
- ... implement a Map Reduce algorithm using the Command Line.
- ... set-up a Docker container and know why we use them for this course.

You will also become familiar (if you aren't already) with `defaultdict`, `re` and `time` in Python, linux piping and sorting, and Jupyter magic commands `%writefile` and `%timeit`. Please refer to the README for detailed submission instructions.

**IMPORTANT:** If you're not familiar with linux, you should read the following tutorial regarding piping and redirecting: <https://ryantutorials.net/linuxtutorial/piping.php> You will need to understand the differences to answer some of the later questions.

**2. Notebook Set-Up**

Before starting your homework run the following cells to confirm your setup.

```
[ ]: # confirm you are running Python 3
import sys
sys.version_info
```

```
[ ]: # imports
import re
```

Saving completed

Mode: Command Ln 1, Col 1 hw1\_Workbook.ipynb

## Question 27

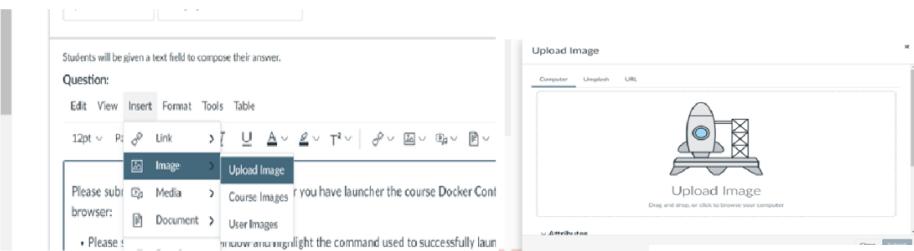
Please upload a copy of the Jupyter notebook in .ipynb format that you used here.

Upload

# HomeWork Assignments: Code Submission Guidelines

For HW questions that require for code snippets, please submit a screenshot of the code cell from your Jupyter notebook using the guidelines provided.

- Please make sure to **submit screenshots of code cells with line numbers on**.
  - To enable line numbers within a Jupyter notebook:
    - Click **View --> Show Line Numbers** in your Jupyter Lab.
    - Line numbers are mandatory and **zero points** can be awarded if line numbers are missing.
- **To upload an image as part of a HW answer please follow the steps outlined here:**
  - Click on Insert -> Image -> Upload Image -> Drag and drop or browse your computer:



## Guidelines for creating screenshots

- Please make sure that the image is properly uploaded and you can **VIEW** it the screenshots are missing.
- To enable line numbers within Jupyter:
  - Click **View --> Show Line Numbers** in your Jupyter Lab Notebook.
  - Line numbers are mandatory and **zero points** can be awarded if line

```
1 #=====
2 #          Your code starts here          #
3 #=====#
4 #
5 #
6 # TODO:
7 def step(x, eta):
8     update = eta * .... #Insert f prime here
9     return x - update
10
11 step(x=..., eta=...)
12
13 #=====
14 #          Your code ends here           #
15 #=====
```

```
1
2 =====#
3 #          Your code starts here          #
4 =====#
5
6 # TODO:
7 def step(x, eta):
8     update = eta * .... #Insert f prime here
9     return x - update
10
11 step(x=..., eta=...)
12
13 =====#
14 #          Your code ends here           #
15 =====#
```

# HW1 preview (9 questions)

The screenshot shows a Jupyter Notebook interface with two panes. The left pane displays a table of contents for 'HW1\_WORKBOOK.ipynb' with 12 numbered sections. The right pane shows the content for section 1.

**Table of Contents (Left Pane):**

- 1. HW1 - Intro to the Map Reduce Paradigm
- 2. Notebook Set-Up
- 3. Question 1: Introductions
- 4. Question 2: Bias - Variance
- 5. Question 3: Tokenizing
- 5.0.1. Q3 Student Answers:
- 6. Load the Data
- 7. Question 4: Word Count in Python
- 7.0.1. Q4 Student Answers:
- 8. Question 5: Unix Sorting
- 8.0.1. Q5 Student Answers:
- 9. Question 6: Parallel Word Count (part 1)
- 9.0.1. Q6 Student Answers:
- 10. Question 7: Parallel Word Count (part 2)
- 10.0.1. Q7 Student Answers:
- 11. Question 8: Scalability Considerations
- 11.0.1. Q8 Student Answers:
- 12. Question 9: Embarrassingly Parallel
- 12.0.1. Q9 Student Answers:

**Section 1 Content (Right Pane):**

## 1. HW1 - Intro to the Map Reduce Paradigm

MIDS w261: Machine Learning at Scale | UC Berkeley School of Information | Spring 2018

Welcome to Machine Learning at Scale! This first homework assignment introduces one of the core strategies in distributed processing: divide and conquer. We'll use the simplest of tasks, word counting, to illustrate the difference between a scalable and non-scalable algorithm. You will be working with the text of *Alice in Wonderland* to put these ideas into practice using Python and Bash scripting. By the end of this week you should be able to:

- ... describe the Bias-Variance tradeoff as it applies to Machine Learning.
- ... explain why we consider word counting to be an "Embarrassingly Parallel" task.
- ... estimate the runtime of embarrassingly parallel tasks using "back of the envelope" calculations.
- ... implement a Map Reduce algorithm using the Command Line.
- ... set-up a Docker container and know why we use them for this course.

You will also become familiar (if you aren't already) with `defaultdict`, `re` and `time` in Python, linux piping and sorting, and Jupyter magic commands `%writefile` and `%timeit`. Please refer to the [README](#) for detailed submission instructions.

**IMPORTANT:** If you're not familiar with linux, you should read the following tutorial regarding **piping** and **redirecting**: <https://ryantutorials.net/linuxtutorial/piping.php> You will need to understand the differences to answer some of the later questions.

## 2. Notebook Set-Up

Before starting your homework run the following cells to confirm your setup.

```
[ ]: # confirm you are running Python 3
import sys
sys.version_info
```

```
[ ]: # imports
import re
```

Saving completed Mode: Command ↵ Ln 1, Col 1 hw1\_Workbook.ipynb

# Short response versus code questions: Q3

The image shows a Jupyter Notebook interface with two main panes. The left pane, titled 'HW1\_WORKBOOK.ipynb', contains a table of contents for a series of questions. The right pane, titled 'Q3 Tasks:', contains three short response and one code-related question.

**Table of Contents (Left Pane):**

- 2. Notebook Set-Up
- 3. Question 1: Introductions
- 4. Question 2: Bias - Variance
- 5. Question 3: Tokenizing
- 5.0.1. Q3 Student Answers:
- 6. Load the Data
- 7. Question 4: Word Count in Python
- 7.0.1. Q4 Student Answers:
- 8. Question 5: Unix Sorting
- 8.0.1. Q5 Student Answers:
- 9. Question 6: Parallel Word Count (part 1)
- 9.0.1. Q6 Student Answers:
- 10. Question 7: Parallel Word Count (part 2)
- 10.0.1. Q7 Student Answers:
- 11. Question 8: Scalability Considerations
- 11.0.1. Q8 Student Answers:
- 12. Question 9: Embarrassingly Parallel
- 12.0.1. Q9 Student Answers:
- 12.0.2. Congratulations, you have completed HW1! Please refer to the readme for

**Q3 Tasks: (Right Pane)**

- a) short response:** In the Naive Bayes algorithm (which we'll implement next week), we'll estimate the *likelihood* of a word by counting the number of times it appears and dividing by the size of the vocabulary (total number of unique words). Using the text: "Alice had an adventure that took alice to wonderland", give a concrete example of how two different tokenizers could cause us to get two different results on this calculation. [ HINT : \_you should not need to read up on Naive Bayes to answer this question\_]
- b) short response:** When tokenizing in this assignment we'll remove punctuation and discard numerical digits by making everything lowercase and then capturing only consecutive letters a to z. Suppose `tokenize(x)` is a Python function that performs the desired tokenization. What would `tokenize("By-the-bye, what became of Alice's 12 hats?!")` output? Type the answer in the space provided below.
- c) code:** Fill in the regular expression pattern in the cell labeled `part_c` so that the subsequent call to `re.findall(RE_PATTERN, ...)` returns the tokenization described above. [ HINT : \_we've taken care of the lowercase part for you. If regex is new to you, check out the `re` documentation or this PyMOTW tutorial.\_]

**5.0.1. Q3 Student Answers:**

- a)** Type your answer here!
- b)** Type your answer here!

```
[ ]: # Part C - Fill in the regular expression
RE_PATTERN = re.compile("")  
***
```

```
[ ]: # Tokenize - DO NOT MODIFY THIS CELL, just run it as is to check your pattern
words = re.findall(RE_PATTERN, "By-the-bye, what became of Alice's 12 hats?!".lower())
print(words)
***
```

# Sample Solution to Q3

HW1\_WORKBOOK.ipynb

- 2. Notebook Set-Up
- 3. Question 1: Introductions
- 4. Question 2: Bias - Variance
- 4.0.1. --- SOLUTION ---
- 5. Question 3: Tokenizing**
- 5.0.1. Q3 Student Answers:
- 5.0.2. --- SOLUTION ---
- 6. Load the Data
- 7. Question 4: Word Count in Python
- 7.0.1. Q4 Student Answers:
- 7.0.2. --- SOLUTION ---
- 8. Question 5: Unix Sorting
- 8.0.1. Q5 Student Answers:
- 8.0.2. --- SOLUTION ---
- 9. Question 6: Parallel Word Count (part 1)
- 9.0.1. Q6 Student Answers:
- 9.0.2. --- SOLUTION ---
- 10. Question 7: Parallel Word Count (part 2)
- 10.0.1. Q7 Student Answers:

10.0.2. --- SOLUTION ---

Labs X PyTorch X Lab-1 X Unit1 X Phasor X Phasor X Bosc X Linea X HW0 X hw1\_1 X hw1\_1 X map3 X

output? type the answer in the space provided below.

• **c) code:** Fill in the regular expression pattern in the cell labeled `part_c` so that the subsequent call to `re.findall(RE_PATTERN, ...)` returns the tokenization described above. [ HINT : we've taken care of the lowercase part for you. If regex is new to you, check out the `re` documentation or this PyMOTW tutorial.]

**5.0.1. Q3 Student Answers:**

a) Type your answer here!

b) Type your answer here!

**5.0.2. --- SOLUTION ---**

a) Different tokenizers might cause certain raw words to get counted as the same or as distinct tokens which would affect both the total number of (unique) words and the individual word counts for those words. For example, suppose we had the following small corpus: "Alice had an adventure that took alice to wonderland" ... a tokenizer that doesn't lowercase words would calculate the likelihood of 'alice' to be  $\frac{1}{9}$  whereas if our tokenizer did lowercase all words the likelihood of 'alice' would be  $\frac{2}{8}$

b) `['by', 'the', 'bye', 'what', 'became', 'of', 'alice', 's', 'hats']`

```
[4]: # Part C - Fill in the regular expression
RE_PATTERN = re.compile("")  
***  
  
[4]: # < ---- SOLUTION ---- >  
# Part C - Fill in the regular expression  
RE_PATTERN = re.compile("[a-zA-Z]+")  
  
[5]: # Tokenize - DO NOT MODIFY THIS CELL, just run it as is to check your pattern
words = re.findall(RE_PATTERN, "By-the-bye, what became of Alice's 12 hats!?" .lower())
print(words)
```

# What this course is and isn't

---

## What the course is about

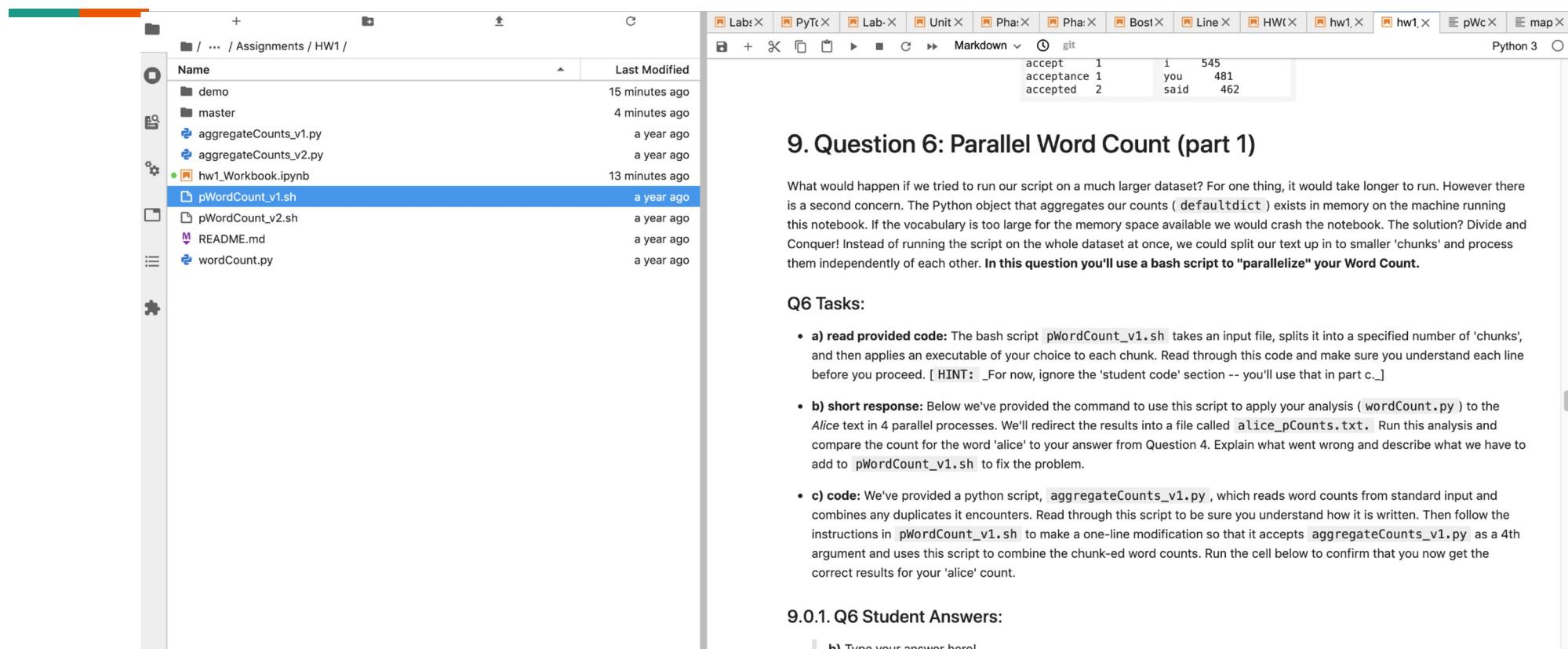
This class is focused on understanding the how and why of implementing large scale machine learning.

- Dive in to the theory of machine learning, unpack equations
- Code up the equations and heuristics on a single CPU
  - and then code them on a cluster of machines
  - Working on a realworld big data machine learning project

## What it isn't about

- Spark in production
- MLOPs
- Computational learning theory
- Hard core data engineering

# Starter code is provided



The screenshot shows a Jupyter Notebook interface. On the left, a file browser displays a directory structure under 'Assignments / HW1 /'. The 'pWordCount\_v1.sh' file is selected. On the right, a code editor shows a Markdown cell with a table containing word counts from the 'alice' text. The table has columns for word, count, and frequency.

word	count	frequency
accept	1	i 545
acceptance	1	you 481
accepted	2	said 462

## 9. Question 6: Parallel Word Count (part 1)

What would happen if we tried to run our script on a much larger dataset? For one thing, it would take longer to run. However there is a second concern. The Python object that aggregates our counts (`defaultdict`) exists in memory on the machine running this notebook. If the vocabulary is too large for the memory space available we would crash the notebook. The solution? Divide and Conquer! Instead of running the script on the whole dataset at once, we could split our text up in to smaller 'chunks' and process them independently of each other. In this question you'll use a bash script to "parallelize" your Word Count.

### Q6 Tasks:

- a) read provided code:** The bash script `pWordCount_v1.sh` takes an input file, splits it into a specified number of 'chunks', and then applies an executable of your choice to each chunk. Read through this code and make sure you understand each line before you proceed. [ HINT: For now, ignore the 'student code' section -- you'll use that in part c...]
- b) short response:** Below we've provided the command to use this script to apply your analysis (`wordCount.py`) to the *Alice* text in 4 parallel processes. We'll redirect the results into a file called `alice_pCounts.txt`. Run this analysis and compare the count for the word 'alice' to your answer from Question 4. Explain what went wrong and describe what we have to add to `pWordCount_v1.sh` to fix the problem.
- c) code:** We've provided a python script, `aggregateCounts_v1.py`, which reads word counts from standard input and combines any duplicates it encounters. Read through this script to be sure you understand how it is written. Then follow the instructions in `pWordCount_v1.sh` to make a one-line modification so that it accepts `aggregateCounts_v1.py` as a 4th argument and uses this script to combine the chunk-ed word counts. Run the cell below to confirm that you now get the correct results for your 'alice' count.

### 9.0.1. Q6 Student Answers:

b) Type your answer here!

## Question 1

5 pts

The Caterpillar and Alice looked at each other for some time in silence: at last the Caterpillar took the hookah out of its mouth, and addressed her in a languid, sleepy voice. "Who are you?" said the Caterpillar.

-- Lewis Carroll, Alice's Adventures in Wonderland, Chapter 4

Tell us about yourself! Briefly describe where you live, how far along you are in MIDS, what other classes you are taking and what you want to get out of w261.

Edit View Insert Format Tools Table

12pt ▾ Paragraph ▾ | **B** *I* U  $\mathcal{A}$   $\mathcal{L}$   $\mathcal{T}^2$  |  $\mathcal{C}$   $\mathcal{G}$   $\mathcal{D}$   $\mathcal{E}$  |  $\mathcal{H}$   $\mathcal{V}$  |  $\mathcal{I}$   $\mathcal{J}$   $\mathcal{K}$  |  $\mathcal{L}$   $\mathcal{M}$   $\mathcal{N}$  |  $\mathcal{O}$   $\mathcal{P}$   $\mathcal{Q}$  |  $\sqrt{x}$   $\phi$

$x^2 +$

G

p

0 words | </> ↵

### Question 1

5 pts

## Equation Editor

x

Basic

Greek

Operators

Relationships

Arrows

Delimiters

Misc



$x^2 +$

G



Directly Edit LaTeX

$x^2 +$

Cancel

Done

p



0 words



43

# Goals of the Course



# Tooling

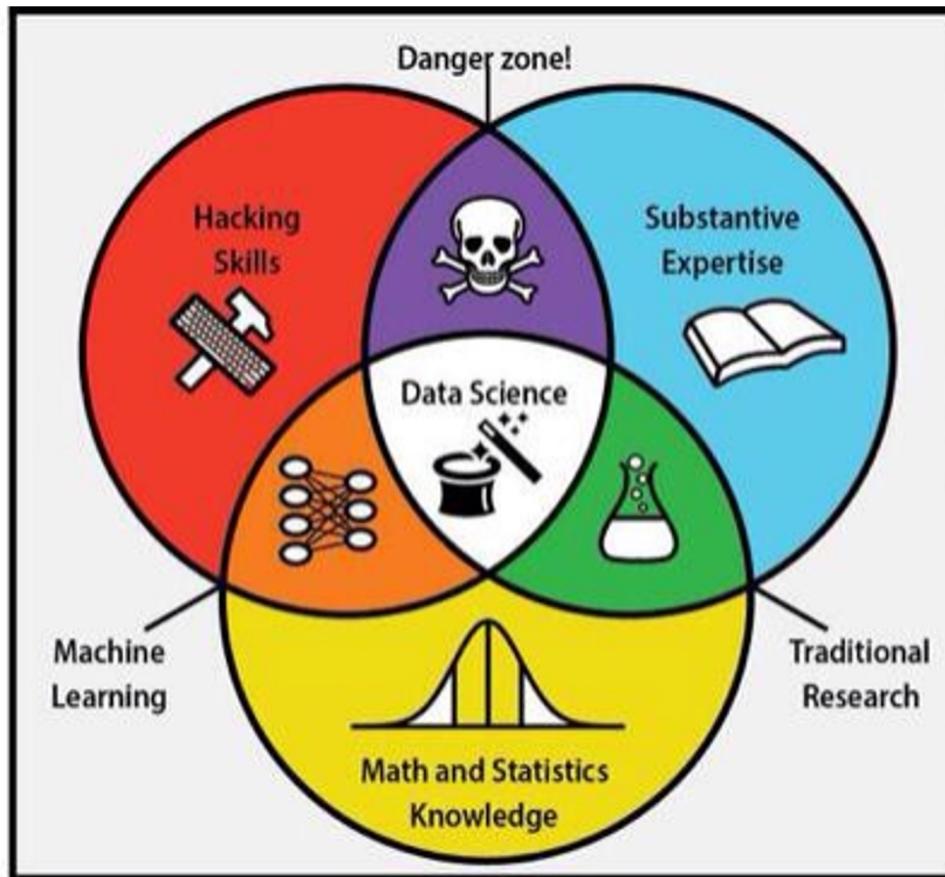
---

- **Hadoop, MapReduce, Spark**

- Where did Hadoop come from?
- Understanding the MapReduce paradigm
- What are the limitations of Hadoop MapReduce?
- Spark as a next generation tool (still MapReduce paradigm)
- How to build machine learning algorithms in Spark (using the Spark operators)

# Data Scientist

---



# Implementing ML algorithms from scratch

---

- Focus on theory, implementation and practice
- **Avoid the “Danger Zone”** (lots of great libraries but....)
- There are several different reasons why implementing algorithms from scratch can be useful:
  - it can help us to understand the inner workings of an algorithm
  - Let's us figure out how to parallelize
  - we can add new features to an algorithm or experiment with different variations of the core idea
  - we want to invent new algorithms or implement algorithms no one has implemented/shared yet
  - we are not satisfied with the API and/or we want to integrate it more "naturally" into an existing software library
  - we could try to implement an algorithm more efficiently
  - we circumvent licensing issues (e.g., Linux vs. Unix) or platform restrictions

# Preview of things to come

---

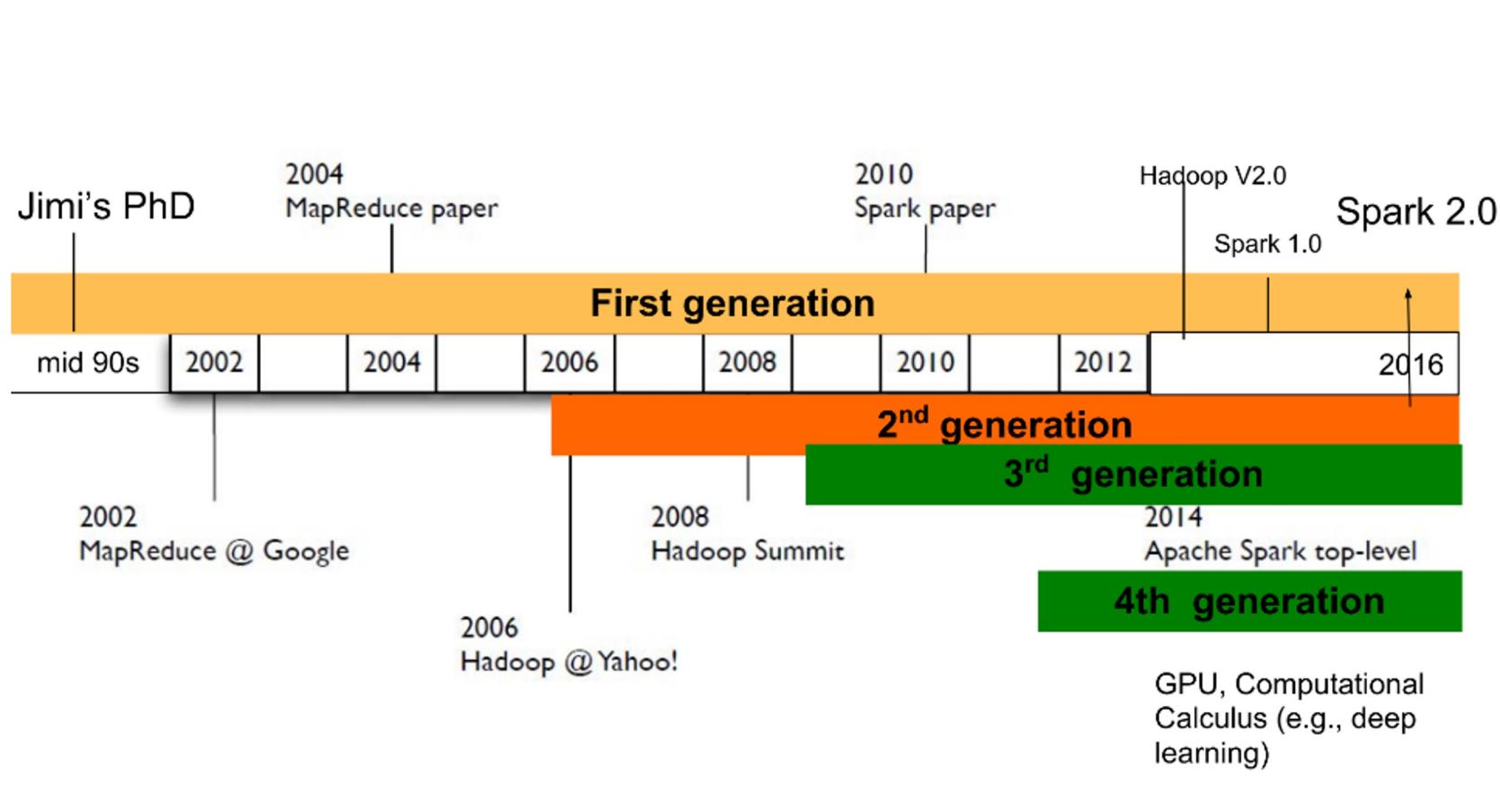
- Parallel frameworks for big data
  - Unix, Hadoop, Spark
- Supervised Machine Learning
  - Convex optimization, gradient descent, linear regression, decision trees, ensembles of models, support vector machines
- Unsupervised
  - Expectation maximization, matrix multiplication, alternating least squares
- Graphs
  - Random walks, PageRank, graph search algorithms such as BFS, shortest path
- Hybrid algos
  - Supervised ML + Random walks
- Applications
  - Digital advertising, social media, healthcare, e-commerce, entertainment

# Data sets

---

- Enron SPAM Data (100k articles)
- Google ngram corpus (2.2 GB)
- Wikipedia graph (2 GB)
- \$100 Billion problem CTR prediction (20-30 GB)
- Airlines and Weather - predicting delays (1 TB uncompressed)

# Evolution of MapReduce frameworks



## Other big data frameworks - “High-performance computing”

<https://en.wikipedia.org/wiki/Supercomputer>

**HPC** - High Performance Computing most generally refers to the practice of aggregating computing power in a way that delivers much higher performance than one could get out of a typical desktop computer or workstation in order to solve large problems in science, engineering, or business.

---

**DryadLINQ** - DryadLINQ combines two important pieces of Microsoft technology: the [Dryad](#) distributed execution engine and the .NET Language Integrated Query ([LINQ](#)).

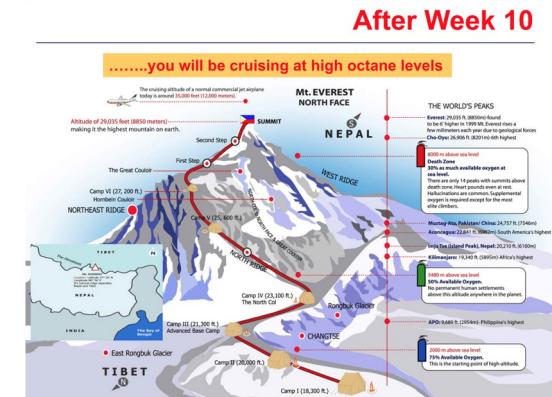
**MPI** - Message Passing Interface (MPI) is a standardized and portable [message-passing](#) standard designed by a group of researchers from academia and industry to function on a wide variety of [parallel computing](#) architectures. The standard defines the syntax and semantics of a core of library routines useful to a wide range of users writing portable message-passing programs in [C](#), [C++](#), and [Fortran](#).

**CUDA** - The CUDA platform, created by NVIDIA, is a software layer that gives direct access to the GPU's virtual [instruction set](#) and parallel computational elements, for the execution of [compute kernels](#).

# Machine Learning @ Scale: “Type II” Fun

- Getting lost, getting cold, getting hungry, getting wet, getting scared, and coming out on top; that's the stuff you remember.
- That's “Type II” fun.
- Type I is the easy, fun-while-it's-happening stuff—mellow powder skiing, lazy cragging, afternoon hiking. You're bummed when it's over, but you'd be hard-pressed to remember more than a few specific examples.
- Type II Tough going, character building and
- Type III fun resides at the other end of the scale—miserable while it's happening, still miserable when it's over and just as miserable to think about later. NOT TO BE CONFUSED WITH “DIFFICULT”.

See more at: <http://www.backcountry.com/explore/type-ii-fun#sthash.CZankeqe.dpuf>



# “Fun” Challenge

---

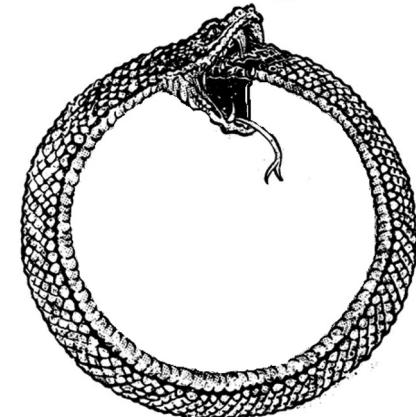
Jane came up with a brilliant new distributed algorithm for solving world hunger. She has a 1TB dataset upon which to train her model. She has just finished writing the last line of her python code! She has provisioned an auto-scaling Spark cluster on Data Proc and she is about to run her algorithm on 1TB of data.

What could possibly go wrong?

(HINT: see Type III fun)

What can Jane do to preserve her sanity?

(HINT: perform sanity checks!)



# Problem solving at Scale

---

1. Local development plus debugging (unit test, systems test)
  1. Benchmark again publicly available libraries
2. Test on the cloud (unit test, systems test)
3. Deploy on the cloud (full scale experiment)

**TEST, TEST, AND TEST AGAIN!!!**

# **Week 1 - “Hello world”**

—

---



# Week 1 Highlights

- Bias Variance Trade-off
- Word count in MapReduce
  - From Word Counts to Multinomial Naïve Bayes
- Embarrassingly Parallel
- Linear Scaling in MapReduce

# bias-variance tradeoff

---

In supervised machine learning, the bias–variance tradeoff is the property of a model that the variance of the parameter estimates across samples can be reduced by increasing the bias in the estimated parameters.

The bias–variance dilemma or bias–variance problem is the conflict in trying to simultaneously minimize these two sources of error that prevent supervised learning algorithms from generalizing beyond their training set:

- Underfitting: The bias error is an error from erroneous assumptions in the learning algorithm. High bias can cause an algorithm to miss the relevant relations between features and target outputs (underfitting).
- Overfitting: The variance is an error from sensitivity to small fluctuations in the training set. High variance can cause an algorithm to model the random noise in the training data, rather than the intended outputs (overfitting).

Finding an  $\hat{f}$  that generalizes to points outside of the training set can be done with any of the countless algorithms used for supervised learning. It turns out that whichever function  $\hat{f}$  we select, we can decompose its [expected](#) error on an unseen sample  $x$  as follows:<sup>[6]:34[7]:223</sup>

$$\mathbb{E}_{D,\varepsilon} \left[ (y - \hat{f}(x; D))^2 \right] = \left( \text{Bias}_D [\hat{f}(x; D)] \right)^2 + \text{Var}_D [\hat{f}(x; D)] + \boxed{\sigma^2}$$

where

$$\text{Bias}_D [\hat{f}(x; D)] = \mathbb{E}_D [\hat{f}(x; D) - f(x)] = \mathbb{E}_D [\hat{f}(x; D)] - \mathbb{E} [y(x)]$$

and

$$\text{Var}_D [\hat{f}(x; D)] = \mathbb{E}_D [(\mathbb{E}_D [\hat{f}(x; D)] - \hat{f}(x; D))^2].$$

The expectation ranges over different choices of the training set  $D = \{(x_1, y_1), \dots, (x_n, y_n)\}$ , all sampled from the same joint distribution  $P(x, y)$  which can for example be done via [bootstrapping](#). The three terms represent:

- the square of the *bias* of the learning method, which can be thought of as the error caused by the simplifying assumptions built into the method. E.g., when approximating a non-linear function  $f(x)$  using a learning method for [linear models](#), there will be error in the estimates  $\hat{f}(x)$  due to this assumption;
- the *variance* of the learning method, or, intuitively, how much the learning method  $\hat{f}(x)$  will move around its mean;
- the irreducible error  $\sigma^2$ .

# Estimating Bias-Variance: 2 Cases

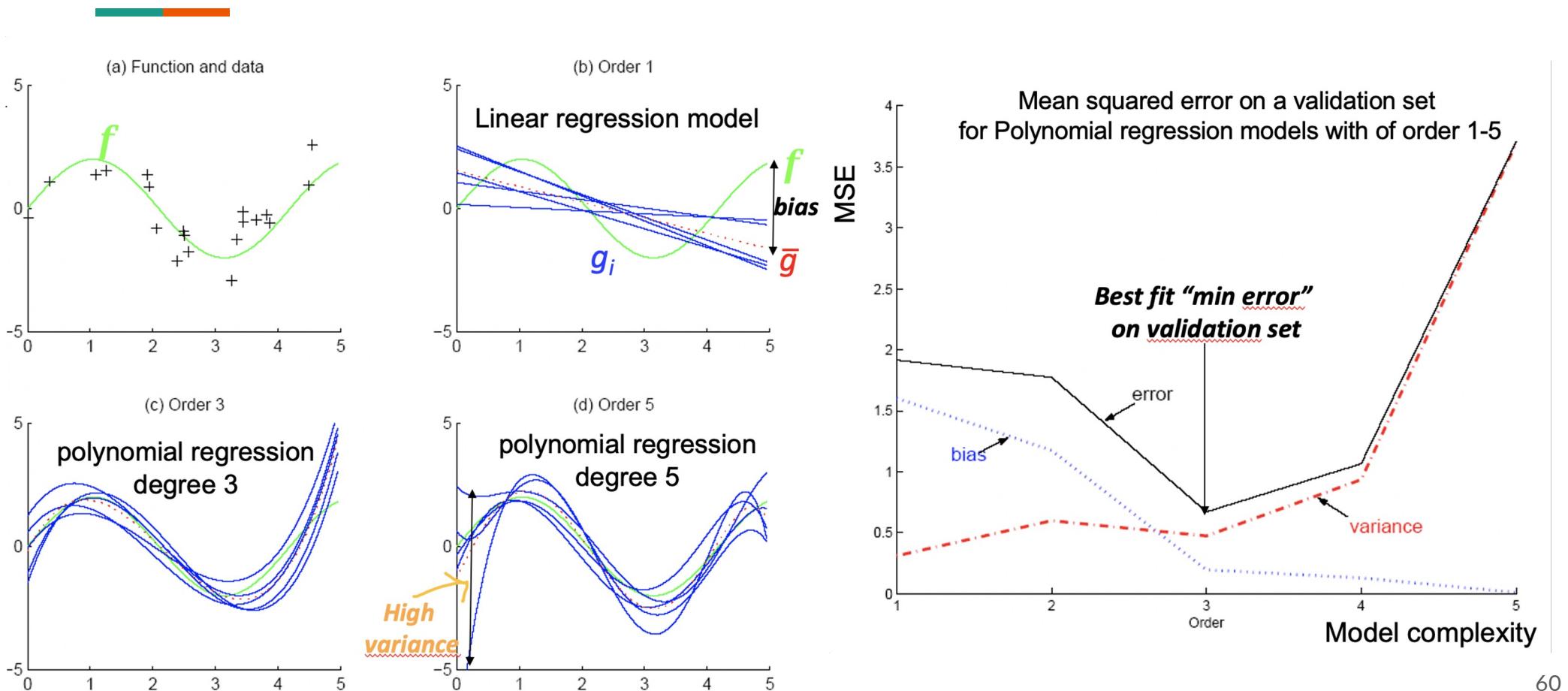
---

- **Case 1: Simulated world: we know the true target function; decompose error (e.g., MSE) into bias, variance, and irreducible error via bootstrap sampling type of analysis**
  - Estimating Bias and Variance in a simulated world where we know the target function
  - We sample training data and train multiple models
  - Then using a validation dataset (a test set is not used in this analysis)
    - Estimate variance of different model predictions
    - Estimate bias (mean prediction of all trained model versus true value)
    - Estimate irreducible error (as we know the true value and the observed noisy value)
- **Case 2: Real world: do NOT know the true target function; decompose error (e.g., MSE) into bias and variance (but not the irreducible error) via bootstrap sampling type of analysis**
  - Estimating Bias and Variance in the real world where we do NOT know the target function
    - Can assume zero noise and estimate bias and variance where the observed target value serves also as the true value

Reference

-<https://blog.insightdatascience.com/bias-variance-tradeoff-explained-fa2bc28174c4>

# Bias-variance decomp for a simulated dataset



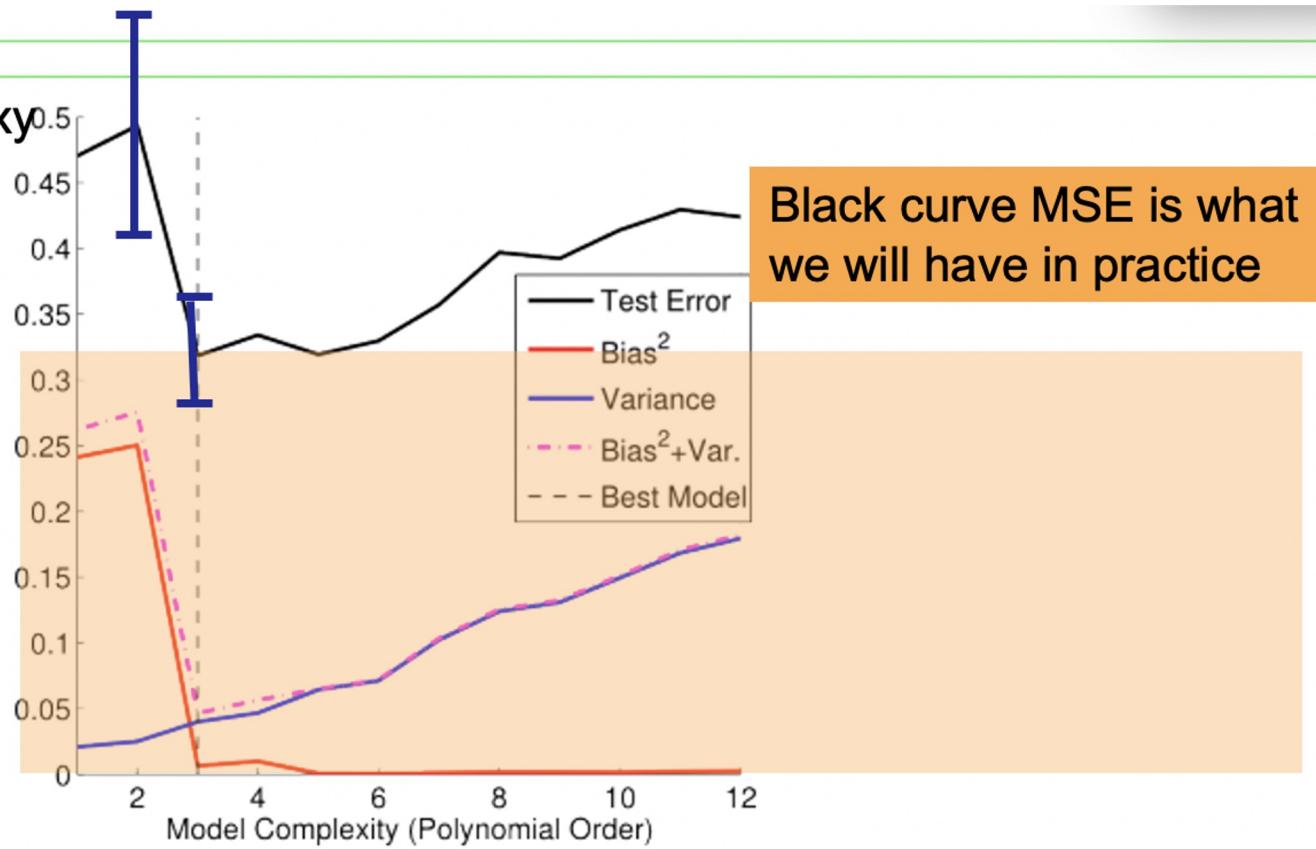
# Bias-Variance Decomposition vs MSE

+ expand source

Black curve is a good proxy  
Bias-variance decomp.

Cross fold validation  
Gives us mean and std

To do a bias-variance  
decomposition  
assume TRUE  
function is available



<http://scott.fortmann-roe.com/docs/BiasVariance.html>

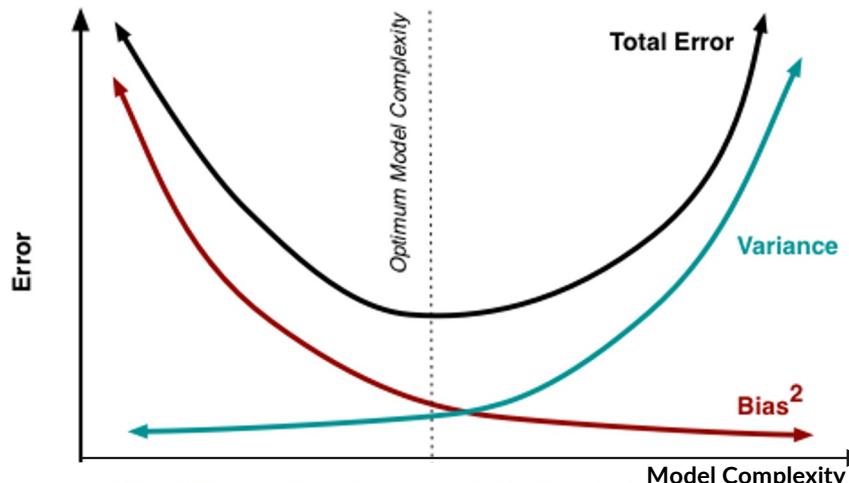


Fig. 6 Bias and variance contributing to total error.

Understanding bias and variance is critical for understanding the behavior of prediction models, but in general what you really care about is overall error, not the specific decomposition. The sweet spot for any model is the level of complexity at which the increase in bias is equivalent to the reduction in variance. Mathematically:

$$\frac{dBias}{dComplexity} = -\frac{dVariance}{dComplexity}$$

# Bias & Variance

---

- Trade-off between bias and variance

What is Bias? (underfitting)

The bias error is an error from erroneous assumptions in the learning algorithm. High bias can cause an algorithm to miss the relevant relations between features and target outputs (underfitting).

What is Variance? (overfitting)

The variance is an error from sensitivity to small fluctuations in the training set. High variance can cause an algorithm to model the random noise in the training data, rather than the intended outputs (overfitting).

# Bias-Variance in lay persons language

---

- BIAS: Assume we have a model that is too simple to explain the data we have (e.g., a linear model when the problem is non-linear).
  - In that case, known as *high bias*, adding more data will not help.
  - high bias models will not benefit from more training examples, but they might very well benefit from more features
- VARIANCE: Assume a model that is too complicated for the amount of data we have. This situation, known as *high variance*, leads to model overfitting.
  - We know that we are facing a high variance issue when the training error is much lower than the test error.
  - High variance problems can be addressed by reducing the number of features, and... yes, by increasing the number of data points.
  - So, no, **more data does not always help**. More features to the rescue!
- Be careful, even Malcolm Gladwell or Chris Anderson get it wrong:
  - **The End of the Scientific Method?**
    - Of course, whenever there is a heated debate about a possible paradigm change, there are people like Malcolm Gladwell or Chris Anderson that make a living out of heating it even more (don't get me wrong, I am a fan of both, and have read most of their books). In this case, Anderson picked on some of Norvig's comments, and misquoted them in an article entitled: [The End of Theory: The Data Deluge Makes the Scientific Method Obsolete](#).

<https://www.kdnuggets.com/2015/06/machine-learning-more-data-better-algorithms.html>

# Quiz

---



How do the following model parameters affect bias and variance?

- K-Nearest Neighbor
  - a. Typically, how does increasing k affect bias and variance?
- Decision trees of depth D
  - a. Typically, how does increasing D affect bias and variance?

Polls

Data at Scale

1. K-Nearset Neighbor. Increasing K...

Increase Bias, Reduce Variance

Decrease Bias, Increase Variance

2. Decision Trees. Increasing D...

Increase Bias, Decrease Variance

Decrease Bias, Increase Variance

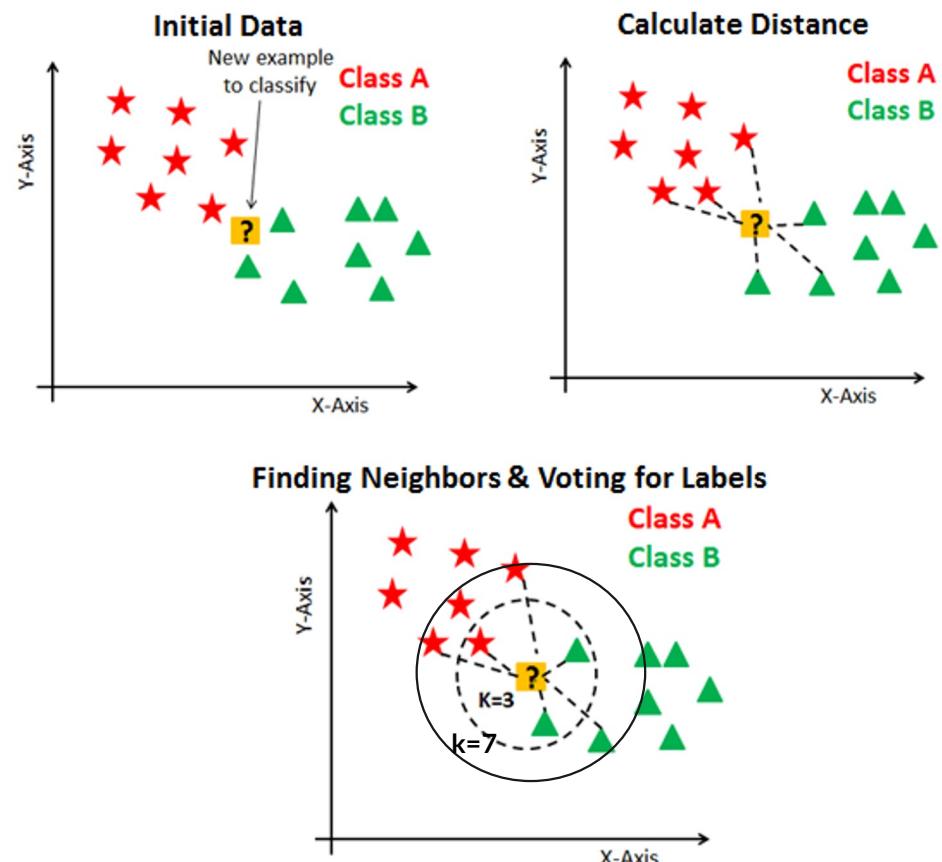
# KNN Classifier with two classes: star versus triangle

large  $k \Rightarrow$  simple model  $\Rightarrow$  underfit  $\Rightarrow$  low variance & high bias (less focus, blurred response)

small  $k \Rightarrow$  complex model  $\Rightarrow$  overfit  $\Rightarrow$  high variance & low bias

When  $k$  increases to inf, the model is simplest. All test data points will belong to the same class: the majority class. This is under-fit, that is, high bias and low variance.

When  $k$  decreases, let's say  $k=1$ , the granularity or resolution is too fine, which is overfit. Overfit  $\Rightarrow$  high variance



# **WORD COUNT**

## **The Hello World of Map reduce**

---

# Word count on a single core machine

---

How would you go about doing a word count on a single core machine?

What data structure in python would you use?

INPUT

```
doc = """
Chinese Beijing Chinese
Chinese Chinese Shanghai
Chinese Macao
Tokyo Japan Chinese
Chinese Chinese Chinese Tokyo Japan.
"""
```

OUTPUT

```
('beijing', 1)
('chinese', 9)
('japan', 2)
('macao', 1)
('shanghai', 1)
('tokyo', 2)
```

# Word count on a single core machine

---

How would you go about doing a word count on a single core machine?

What data structure in python would you use?

```
doc = """  
Chinese Beijing Chinese  
Chinese Chinese Shanghai  
Chinese Macao  
Tokyo Japan Chinese  
Chinese Chinese Chinese Tokyo Japan.  
"""
```

INPUT

```
#version 1  
word_counts = {}  
for word in doc.lower().split():  
    if word in word_counts:  
        word_counts[word] += 1  
    else:  
        word_counts[word] = 1
```

OUTPUT

```
('beijing', 1)  
('chinese', 9)  
('japan', 2)  
('macao', 1)  
('shanghai', 1)  
('tokyo', 2)
```

# What is the problem with this approach if the data is very large?

## Word count on a single machine using a key-value dictionary

[back to top](#)

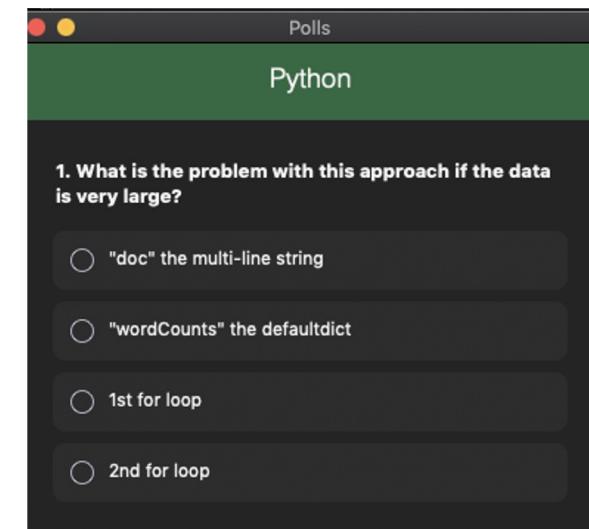
```
# Here is an example of wordcounting with a defaultdict (dictionary structure with a nice
# default behaviours when a key does not exist in the dictionary
import re
from collections import defaultdict

doc = """
Chinese Beijing Chinese
Chinese Chinese Shanghai
Chinese Macao
Tokyo Japan Chinese
Chinese Chinese Chinese Tokyo Japan.
"""

# Initialize a defaultdict, where the default value is an int
wordCounts=defaultdict(int)

# loop over all the words in the doc, and increment the count as you see words
for word in re.findall(r'[a-z]+', doc.lower()):
    wordCounts[word] += 1

# Sort by word, and print the top 10 words with their counts
for key in sorted(wordCounts)[0:10]:
    print (key, wordCounts[key])
```



('beijing', 1)  
(chinese', 9)  
(japan', 2)  
(macao', 1)  
(shanghai', 1)  
(tokyo', 2)

# WordCount in a Colab Notebook

<https://colab.research.google.com/drive/1mPmTWRQ8f4y8QSYgBYd-0T2Qw53hQkf3?usp=sharing>

The screenshot shows a Google Colab notebook interface. The title bar says 'Word\_count\_in\_Python.ipynb'. The menu bar includes File, Edit, View, Insert, Runtime, Tools, Help, and 'All changes saved'. A sidebar on the left has sections for '+ Code' and '+ Text', and a expanded section for 'TASKS' which contains the instruction: 'Complete/debug the following word count code and verify you get the same response as above:'. Below this, there is a code cell containing Python code for a Counter object:

```
Counter({'yesterday': 1,
          'i': 4,
          'went': 1,
          'fishing': 1,
          'dont': 1,
          'fish': 2,
          'that': 2,
          'often': 1,....})
```

Below this is another code cell with the following code:

```
[ ] 1 # WordCount version 1
2 text = """Yesterday I went fishing. I don't fish that often,
3 so I didn't catch any fish. I was told I'd enjoy myself,
4 but it didn't really seem that fun."""
5
6 word_counts = {}
7 for word in text.lower().split():
8     if word in word_counts:
9         word_counts[word] +=1
10    else: word_counts[word] = 1
```

At the bottom of the code cell is a play button icon followed by the text: '1 ## Run the following word count and verify it works correctly'. The final code cell at the bottom is:

```
[ ] 1 from collections import defaultdict
2
3 counter = defaultdict(int)
4 for word in text.lower().split():
5     counter[word] += 1
```

# pdb in a jupyter notebook cell

---

The built-in [Python debugger](#) `pdb` works just fine in a Jupyter notebook. With `import pdb; pdb.set_trace()` we can enter the debugger and get a little interactive prompt.

```
1 def add_to_life_universe_everything(x):
2     answer = 42
3     import pdb; pdb.set_trace()
4     answer += x
5     return answer

add_to_life_universe_everything(12)
```

For more background on `pdb` see  
<https://davidhamann.de/2017/04/22/debugging-jupyter-notebooks/>

In the debugger we can then print our variables, evaluate code to inspect the current stack, etc.

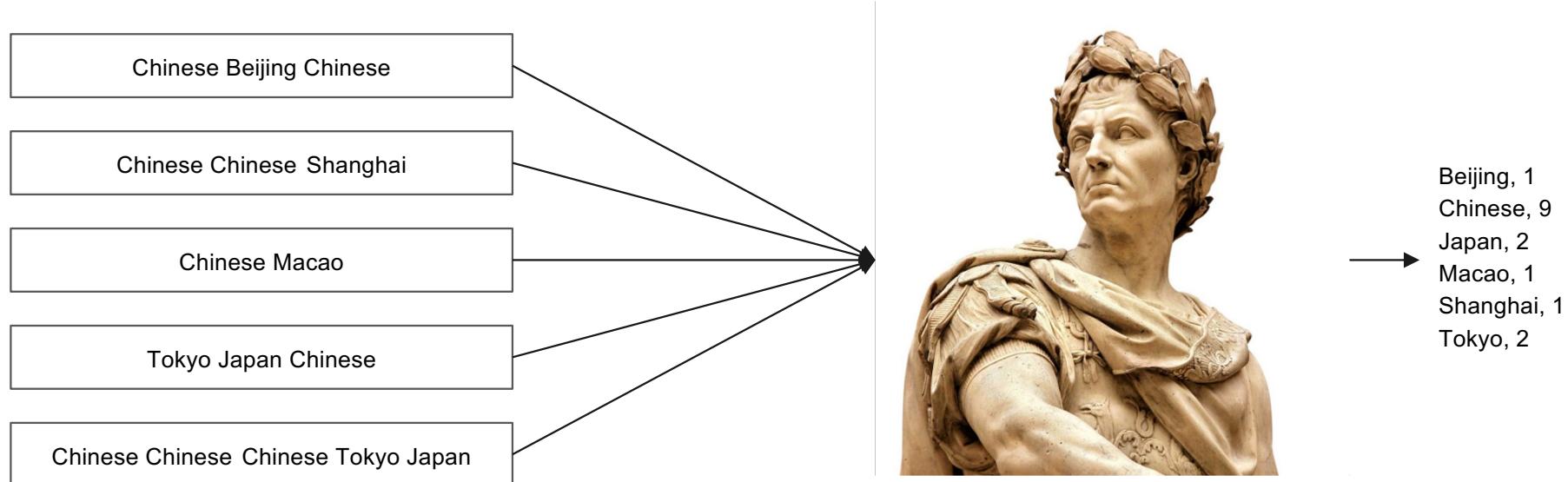
```
> <ipython-input>(4)add_to_life_universe_everything(12)
-> answer += x
(Pdb) answer + 12 - "dummy"
42
(Pdb) n
> <ipython-input>(6)add_to_life_universe_everything(6)
-> return answer
(Pdb) answer
54
(Pdb) c
```

Here, we print `answer`, then execute and go to the next line with `n`, print `answer` again and then continue the program with `c`. **You can get an overview of the commands with `h`, or specifically with `help <command>`.**

# Divide and Conquer (a.k.a. merge-sort) to the rescue!

---

Can we take our big task, and split it up into a bunch of smaller tasks that we can perform independently?



# [https://en.wikipedia.org/wiki/Merge\\_sort](https://en.wikipedia.org/wiki/Merge_sort)



In computer science, **merge sort** (also commonly spelled **mergesort**) is an efficient, general-purpose, **comparison-based sorting algorithm**. Most implementations produce a **stable sort**, which means that the order of equal elements is the same in the input and output. Merge sort is a **divide and conquer algorithm** that was invented by **John von Neumann** in 1945.

6 5 3 1 8 7 2 4

An example of merge sort. First divide the list into the smallest unit (1 element), then compare each element with the adjacent list to sort and merge the two adjacent lists. Finally all the elements are sorted and merged.

**Worst-case performance:**  $O(n \log n)$

**Best-case performance**  $O(n \log n)$  typical,

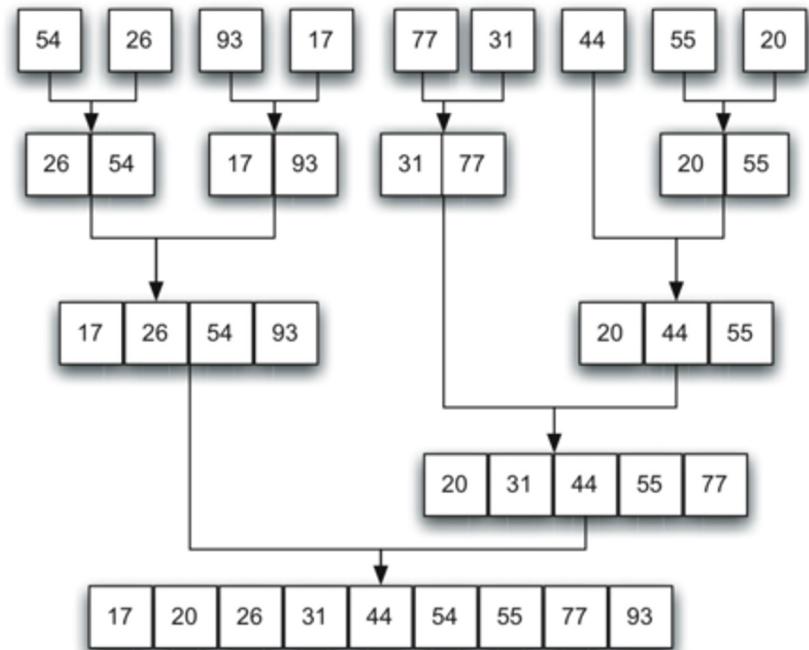
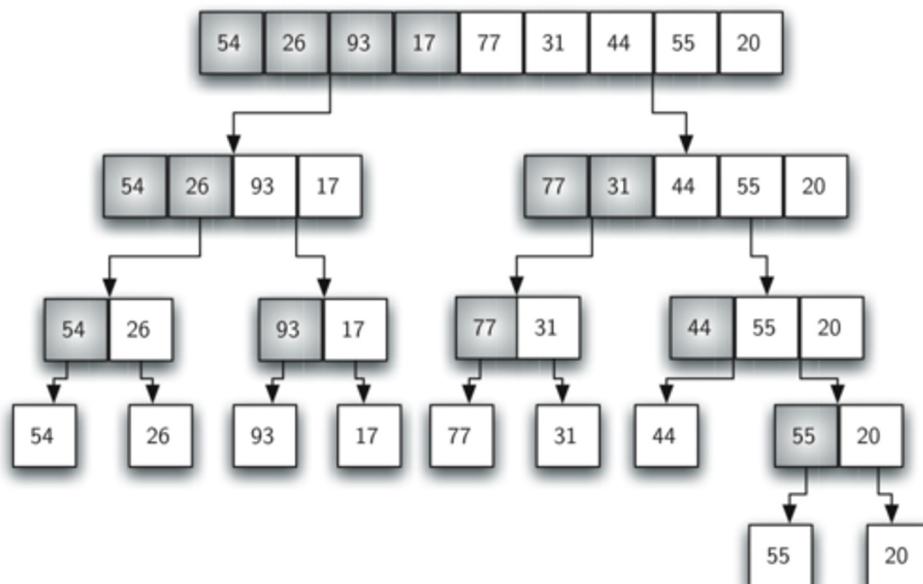
$O(n)$  natural variant

**Average performance:**  $O(n \log n)$

**Worst-case space complexity**

$O(n)$  total with  $O(n)$  auxiliary,  $O(1)$  auxiliary with linked lists [1]

# Merge Sort



[/HelpfulResources/merge-sort.ipynb](#)

<https://www.interviewbit.com/tutorial/merge-sort-algorithm/>

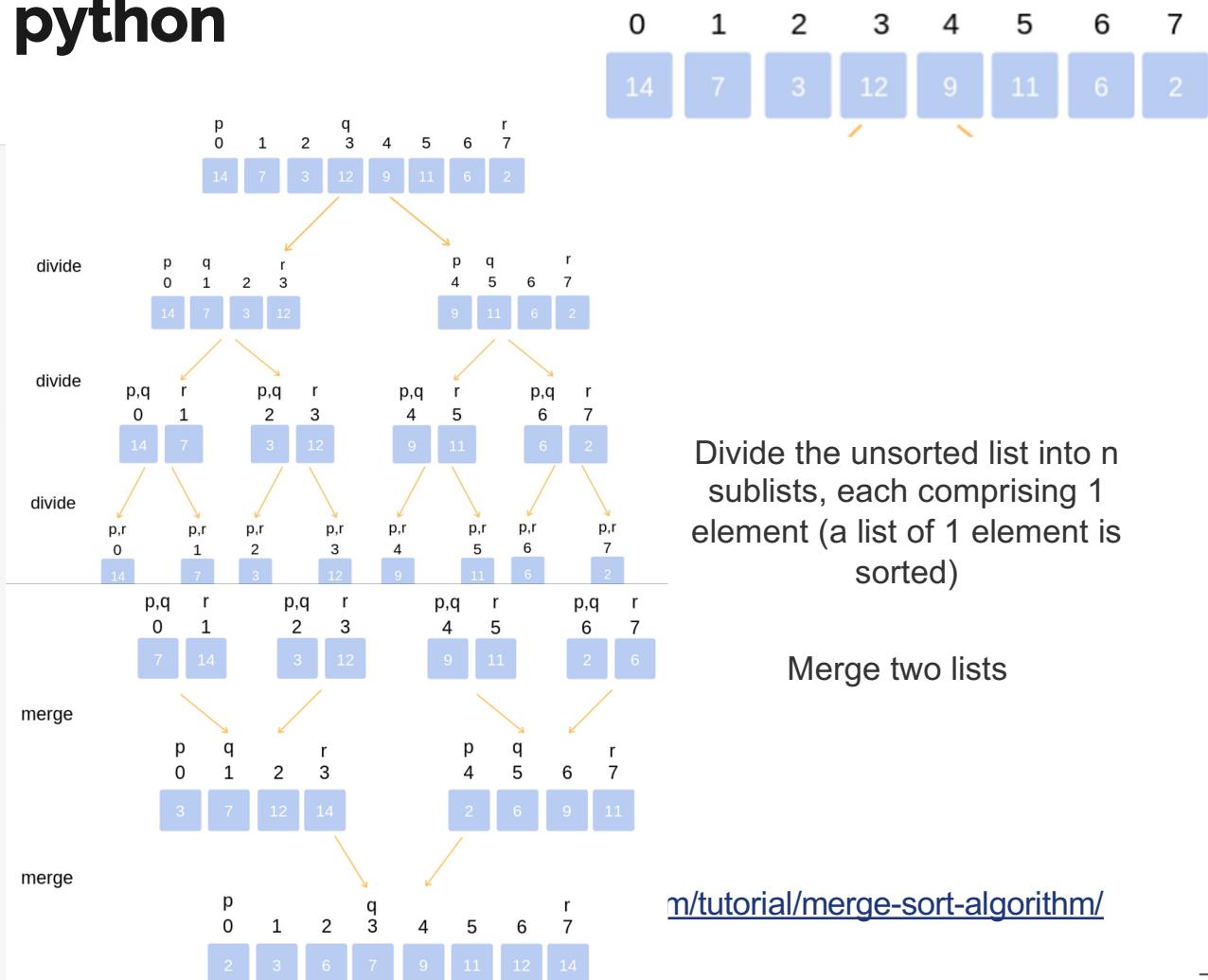
# RECURSIVE merge sort in python

## Top-down implementation

```

1 # example of merge sort in Python (RECURSIVE)
2 # merge function take two intervals
3 # one from start to mid
4 # second from mid+1, to end
5 # and merge them in sorted order
6 def merge(Arr, start, mid, end) :
7
8
9     # create a temp array
10    temp[] = [0] * (end - start + 1)
11
12    # crawlers for both intervals and for temp
13    i, j, k = start, mid+1, 0
14
15    # traverse both lists and in each iteration add smaller of both elements in temp
16    while(i <= mid and j <= end) :
17        if(Arr[i] <= Arr[j]) :
18            temp[k] = Arr[i]
19            k += 1; i += 1
20        else :
21            temp[k] = Arr[j]
22            k += 1; j += 1
23
24    # add elements left in the first interval
25    while(i <= mid)
26        temp[k] = Arr[i]
27        k += 1; i += 1
28
29    # add elements left in the second interval
30    while(j <= end)
31        temp[k] = Arr[j]
32        k += 1; j += 1
33
34    # copy temp to original interval
35    for(i = start; i <= end; i += 1)
36        Arr[i] = temp[i - start]
37
38
39 # Arr is an array of integer type
40 # start and end are the starting and ending index of current interval of Arr
41
42 def mergeSort(Arr, start, end) {
43
44    if(start < end) :
45        mid = (start + end) / 2
46        mergeSort(Arr, start, mid)
47        mergeSort(Arr, mid+1, end)
48        merge(Arr, start, mid, end)

```



Divide the unsorted list into n sublists, each comprising 1 element (a list of 1 element is sorted)

Merge two lists

<https://www.tutorialspoint.com/tutorials/tutorial/merge-sort-algorithm/>

# **TimSort used in Spark (instead of mergesort)**

---

TimSort was born in practice, a hybrid of Merge Sort and Insertion (decrease and conquer) Sort

Tim Sort was first implemented in 2002 by Tim Peters for use in Python. It allegedly came from the understanding that most sorting algorithms are born in schoolrooms, and not designed for practical use on real world data. Tim Sort takes advantage of common patterns in data, and utilizes a combination of:

- Chunk, Runs, and insertion sort
- Merge from mergesort

# Example: Merge Sort in Unix

---

Consider files a.txt and b.txt with following sorted content.

Then one can merge the two files and generate a sorted output using the -m switch

Challenge:

- Tweak the merge-sort command to merge sort on the second field.
- Does the output change?

```
cat a.txt      #show contents of sorted file  
KEY    PAYLOAD  
Deep    1  
Jimi    2  
Learning 2
```

```
cat b.txt      #show contents of sorted file  
Amil    1  
Jimi    3
```

```
$sort -m a.txt b.txt  #merge sort the two files  
Amil    1  
Deep    1  
Jimi    2  
Jimi    3  
Learning 2
```

# Sort command in linux with examples (sorting is key in map-reduce)



## Sorting by a Single Column

Sorting by single column requires the use of the `-k` option. You must also specify the start column and end column to sort by. When sorting by a single column, these numbers will be the same. Here is an example of sorting a CSV (comma delimited) file by the second column.

```
#sorting a CSV by the values in column 2
sort -k2,2 -t ',' sample.txt
```

## Sorting by Multiple Columns: contiguous versus non-contiguous

Sorting by multiple columns is similar to sorting by a single column. To sort on a range of columns, simply specify the start and end columns in the column range to use for sorting. Here is an example of sorting a CSV on columns 2 through 4.

```
#sorting a CSV by the values in columns 2-4
sort -k2,4 -t ',' sample.txt
```

To sort on a **non-contiguous set of columns**, you must use the `-k` option multiple times. Here is an example of sorting by column 2 then column 6:

```
#sorting a CSV by the values in column 2 then column 6
sort -k2,2 -k6,6 -t ',' sample.txt
```

For details please see: <https://phpfog.com/linux-sort-command-examples/>

# Unix sort examples

sort -t"," -k1,1    sort -t"," -k2,2nr    sort -t"," -k1,1 -k2,2nr

Sort by field 1 (default alphabetically), delimiter ","

`cat unix-sort-example.txt | sort -t"," -k1,1 unix-sort-example.txt`

`cat unix-sort-example.txt | sort -t"," -k1,1 unix-sort-example.txt`

Unix,30	AIX,25
Solaris,10	HPUX,100
Linux,25	Linux,20
Linux,20	Linux,25
HPUX,100	Solaris,10
AIX,25	Unix,30

Sort by field 2 numerically reverse, delimiter ","

`cat unix-sort-example.txt | sort -t"," -k2,2nr unix-sort-example.txt`

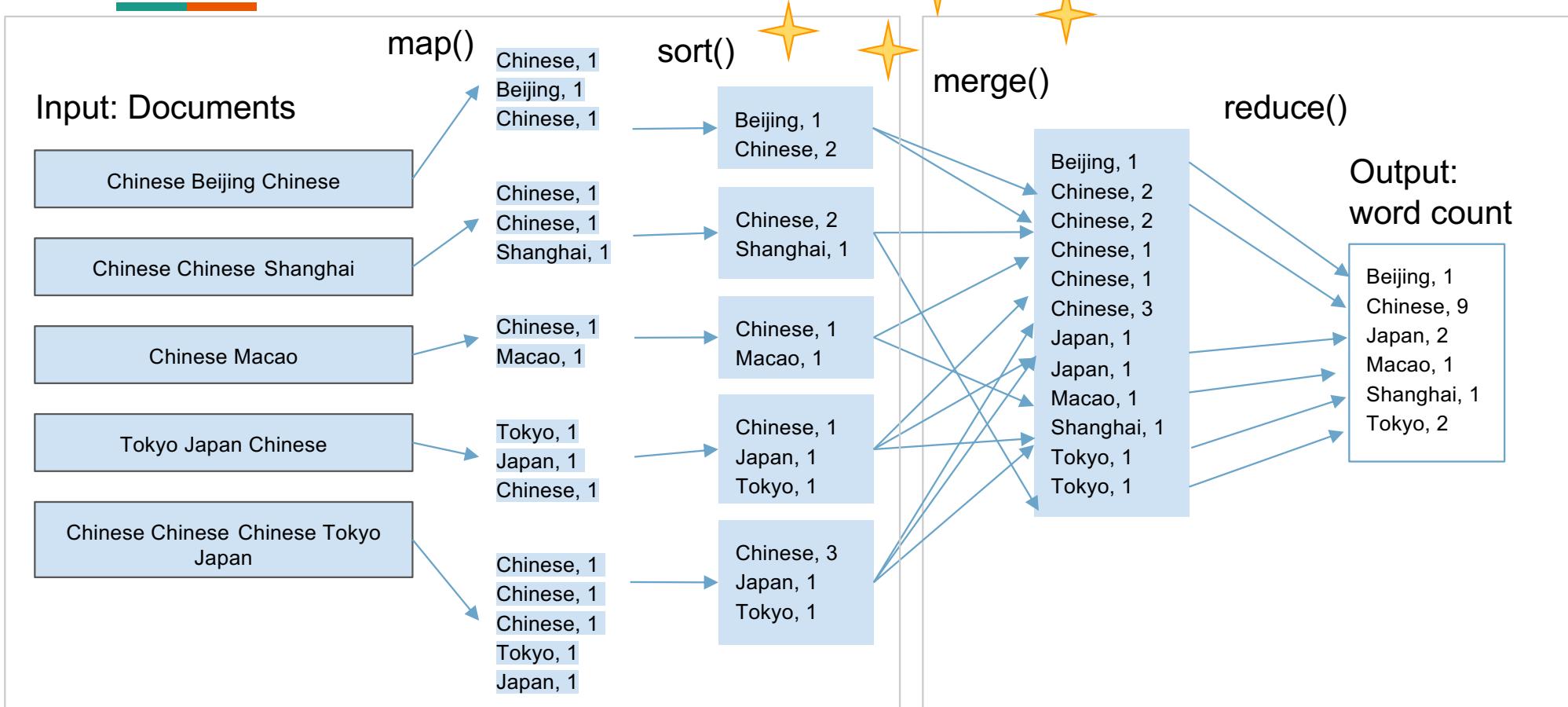
Unix,30	HPUX,100
Solaris,10	Unix,30
Linux,25	AIX,25
Linux,20	Linux,25
HPUX,100	Linux,20
AIX,25	Solaris,10

Sort by field 1 alphabetically first, then by field 2 numeric reverse

`cat unix-sort-example.txt | sort -t"," -k1,1 -k2,2nr unix-sort-example.txt`

Unix,30	AIX,25
Solaris,10	HPUX,100
Linux,25	Linux,25
Linux,20	Linux,20
HPUX,100	Solaris,10
AIX,25	Unix,30

# Single Reducer example



# MapReduce - Key Concepts

---

## Phases of MapReduce

- Map phase
  - Each mapper takes a line as input and breaks it into words. It then emits a key/value pair of the word and 1.
- Shuffle/sort phase (*partition and sort the mapper output*)
  - The output from the mappers is sorted by key and shuffled into the reducers such that key/value pairs with the same key end up in the same reducer.
  - Sort is a simple and very useful command found in Unix systems. It rearranges lines of text numerically and/or alphabetically. (*Hadoop Streaming KeyBasedComparator is modeled after Unix sort, and utilizes command line options which are the same as Unix sort command line options*).
  - <http://www.theunixschool.com/2012/08/linux-sort-command-examples.html>
- Reduce phase
  - Each reducer sums the counts for each word and emits a single key/value with the word and sum.



# Embarrassingly parallel

---

In parallel computing, an **embarrassingly parallel** workload or problem (also called **perfectly parallel**, **delightfully parallel** or **pleasingly parallel**) is one where little or no effort is needed to separate the problem into a number of parallel tasks.<sup>[1]</sup> This is often the case where there is little or no dependency or need for communication between those parallel tasks, or for results between them.<sup>[1]</sup>

No shared state

- a `map()` task does not need to know about any other map task
- a `reduce()` task has all the context it needs to aggregate for any particular key, it does not share any state with other reduce tasks.

Taken as a whole, this design means that the stages of the pipeline can be easily distributed to an arbitrary number of machines.

Workflows requiring massive datasets can be easily distributed across hundreds of machines because there are no inherent dependencies between the tasks requiring them to be on the same machine (ie, sub tasks never need to communicate).

# Test your hacking skills

---

- Linux command line
  - e.g., a sample task could be count the number lines in a file (wc); sort a file based on the second field in reverse numerical order; extract the second field from a TSV file
- Python dictionaries
  - e.g., implement word count by themselves
- Regular expressions
  - e.g., find all the phone numbers in a chunk of text, 415-630-0893, (415)-6300893, 4156300893
- Debugging in Jupyter notebooks using pdb
- STRETCH TASK: Numpy basics for vector multiplication/addition, matrix initialization, matrix multiplication
  - e.g., randomly define a 4x3 matrix, and a 3x4 matrix and multiply them.
- STRETCH TASK: review sorting algos
  - e.g., mergesort and possibly implement it

# From Word Counts to Multinomial Naïve Bayes



Next Week:

How can we use what we learned from word count to implement a distributed Naive Bayes?

# End of lecture

**Feel free to contact me at James.Shanahan \_\_AT\_\_ gmail.com**