Lecture 2: Data Science Concepts and MapReduce Programming Model

COSC 526: Introduction to Data Mining

THE UNIVERSITY OF TENNESSEE KNOXVILLE

Live Chat:
What we learned from 5 million books

Discussions

- **Jean-Baptiste Michel and Erez Lieberman Aiden** tell us about "**What we learned from 5 million books**"

https://www.ted.com/talks/jean_baptiste_michel_erez_lieberman_aiden_what_we_learned_from_5_million_books

Answer these questions related to the talk:

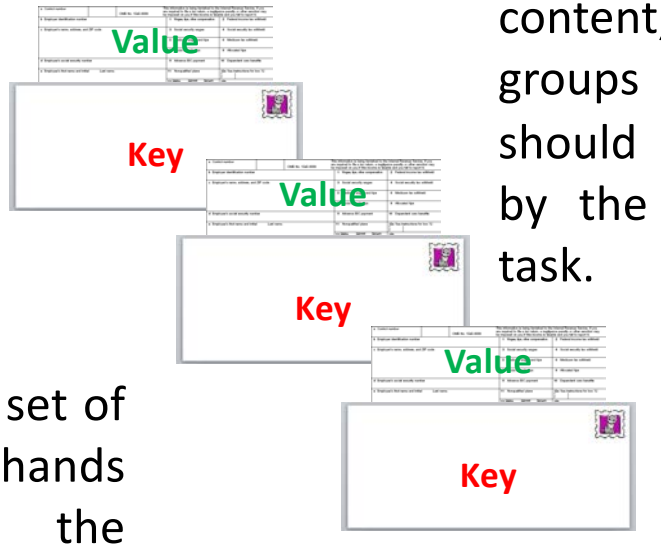
- What is the take-away of this talk? Summarize it in up to 3 sentences.
- What are metadata?
- What is a n-gram?
- What is the suppression index?
- What is culturomics?

The Canonical MapReduce Programming Model

The MapReduce Workflow

Step 1: Map

The **map** task creates a set of **Key-Value** pairs and hands them off to the **sorting**/**shuffling** task.

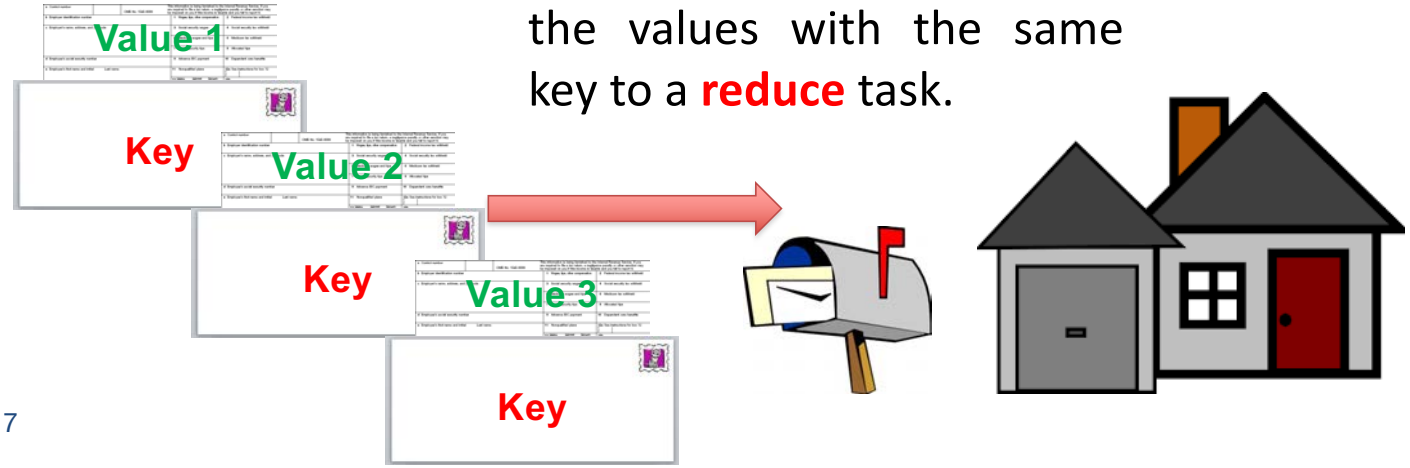


The **Value** is the content, the **Key** groups content that should be processed by the same **reduce** task.

The MapReduce Workflow

Step 2: Sort/Shuffle

The **sorting** and **shuffling** tasks collect all the **key-value** pairs and deliver all the values with the same key to a **reduce** task.

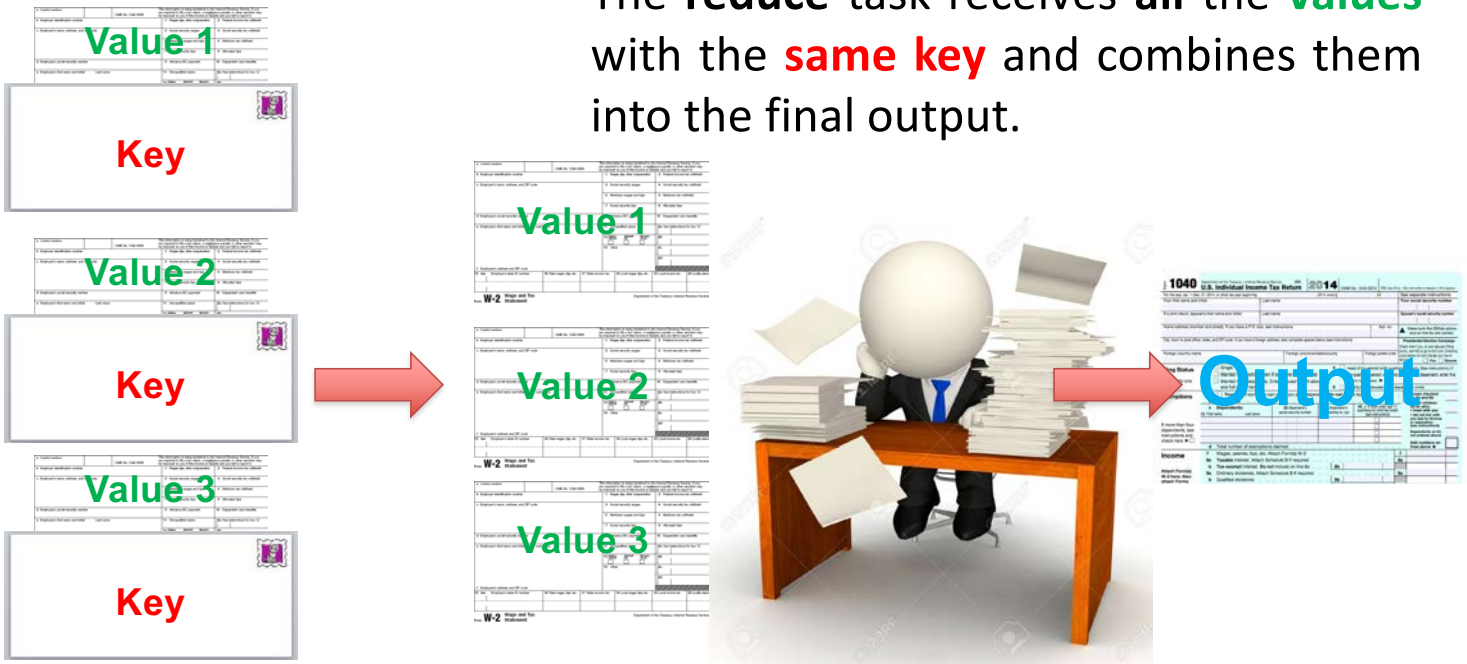


7

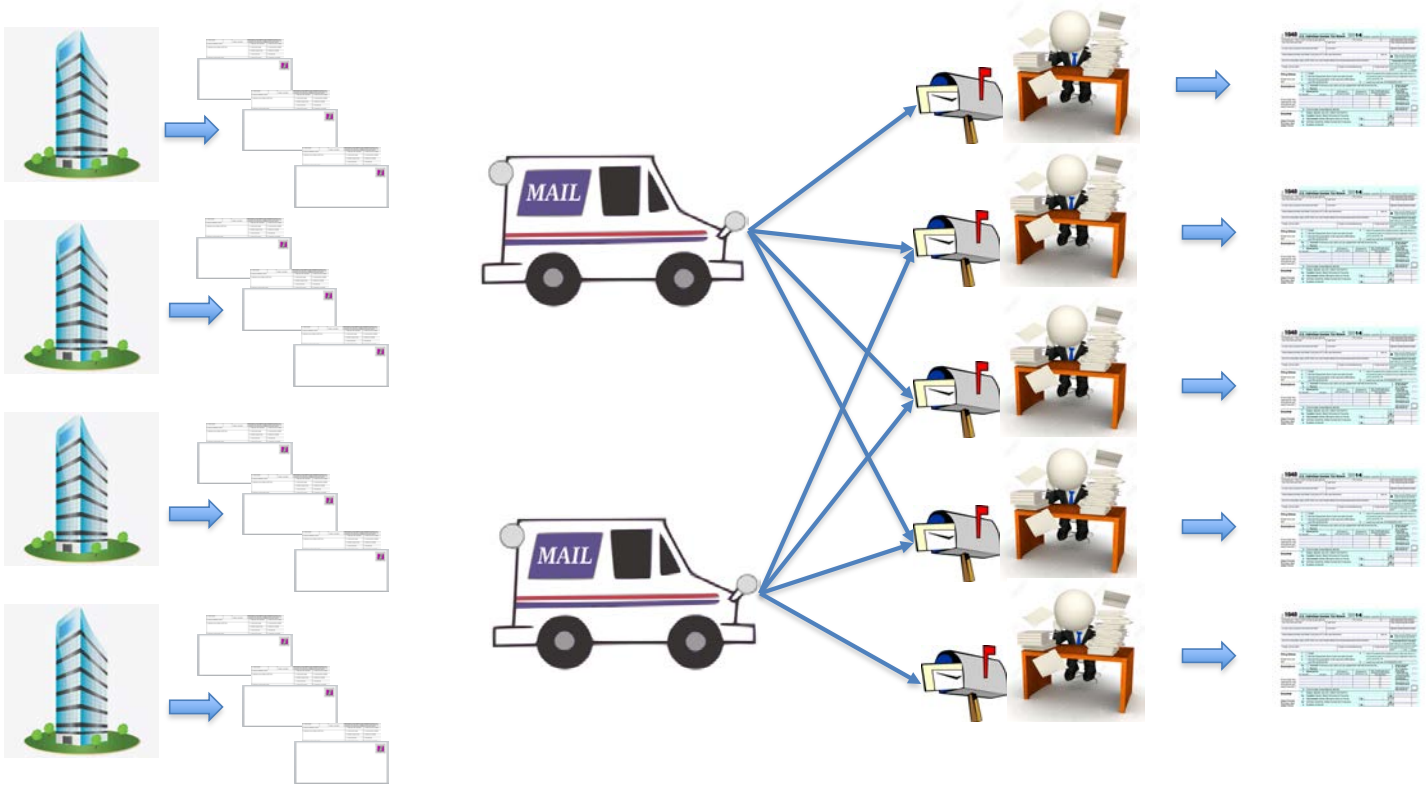
The MapReduce Workflow

Step 3: Reduce

The **reduce** task receives **all** the **values** with the **same key** and combines them into the final output.



The MapReduce Workflow



Map

- The map function is implemented by the user.
- Each map task processes a single line of an input file at a time.
- Map tasks do not communicate with other map tasks.
- Map tasks communicate with reduce tasks **only** through the content of the value in the KV pair.
- A single map task may emit **any number** of KV pairs (including none).

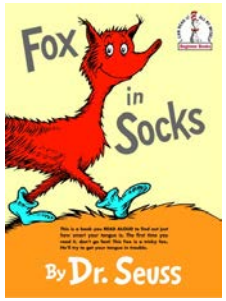
Sort/Shuffle

- The sorting and shuffling process is implemented at the framework level and is opaque to the user.

Reduce

- The reduce function is implemented by the user.
- Each reduce task receives **all** the **values** associated with a given **key**.
- Reduce tasks do not communicate with each other
- A single reduce task may emit **any number** of result values for each **key** it processes.

WordCount: an example of MapReduce



Map

Key **Value**

When tweetle beetles fight, → (When, 1), (**tweetle**, **1**), (beetles, 1), (fight, 1)

it's called a tweetle beetle battle. → (it's, 1), (called, 1), (a, 1), (tweetle, 1), (beetle, 1), (battle, 1)

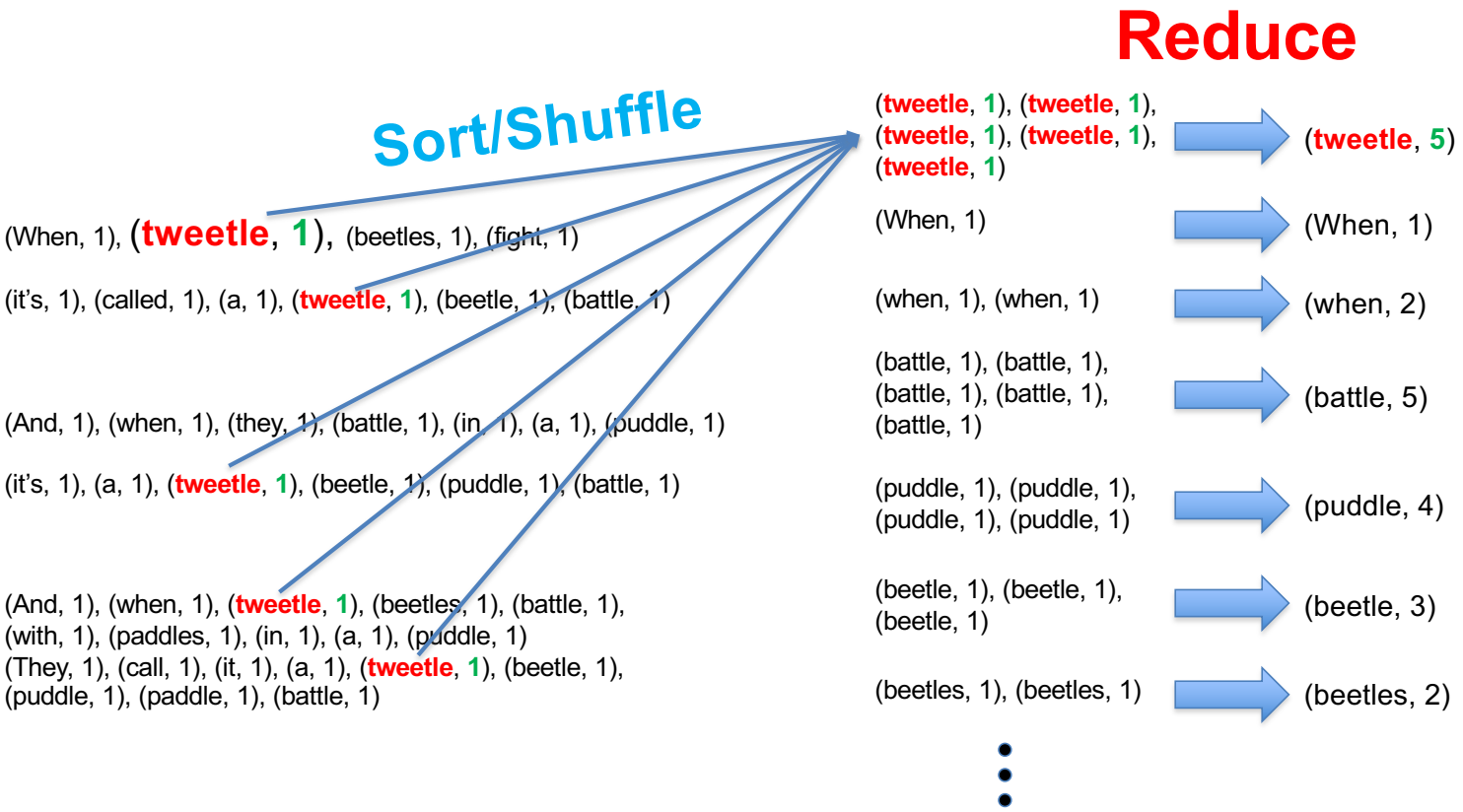
And when they battle in a puddle, → (And, 1), (when, 1), (they, 1), (battle, 1), (in, 1), (a, 1), (puddle, 1)

it's a tweetle beetle puddle battle. → (it's, 1), (a, 1), (tweetle, 1), (beetle, 1), (puddle, 1), (battle, 1)

And when tweetle beetles battle with paddles in a puddle, → (And, 1), (when, 1), (tweetle, 1), (beetles, 1), (battle, 1), (with, 1), (paddles, 1), (in, 1), (a, 1), (puddle, 1)

They call it a tweetle beetle puddle paddle battle. → (They, 1), (call, 1), (it, 1), (a, 1), (tweetle, 1), (beetle, 1), (puddle, 1), (paddle, 1), (battle, 1)

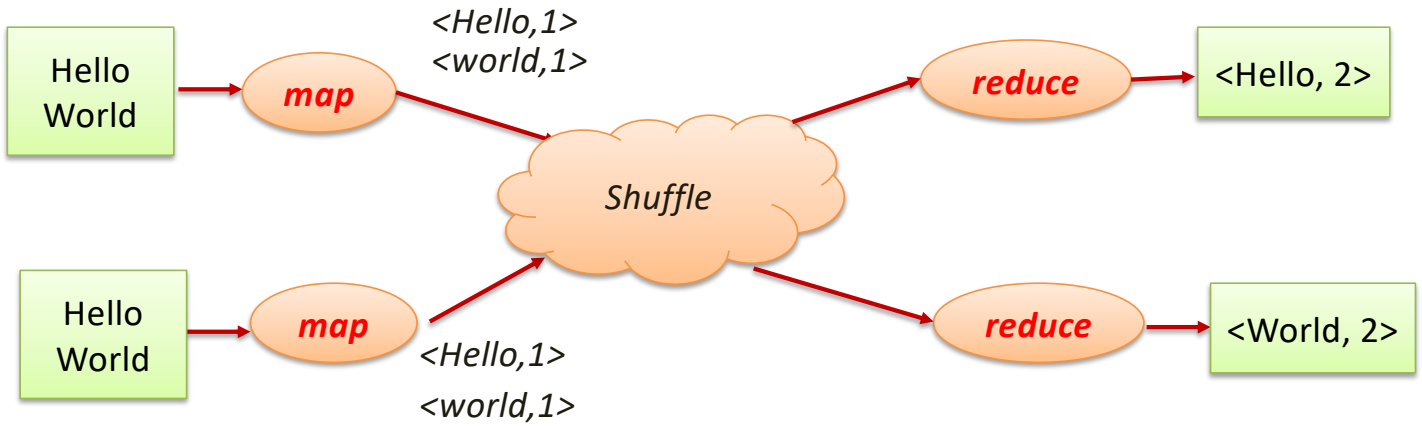
WordCount: an example of MapReduce



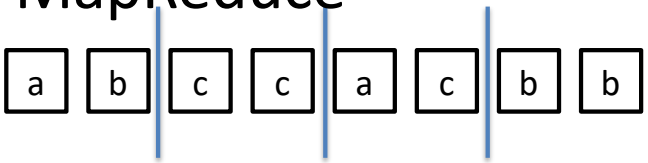
Canonical MapReduce

- MapReduce runtime handles the parallel job execution, communication, and data movement
- Users provide *map* and *reduce* functions

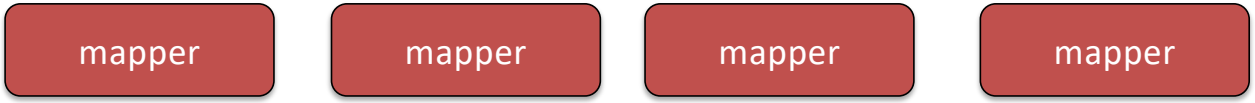
Wordcount example:



The **Canonical** MapReduce



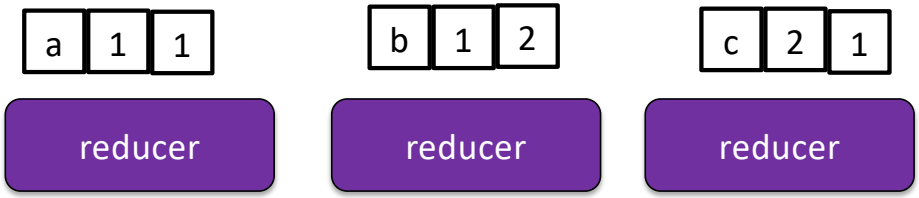
Mappers: applied to all input data



Arbitrary number of key-value pairs

Barrier: distributed sort and group by key

Reducers: applied to all values associated with the same key

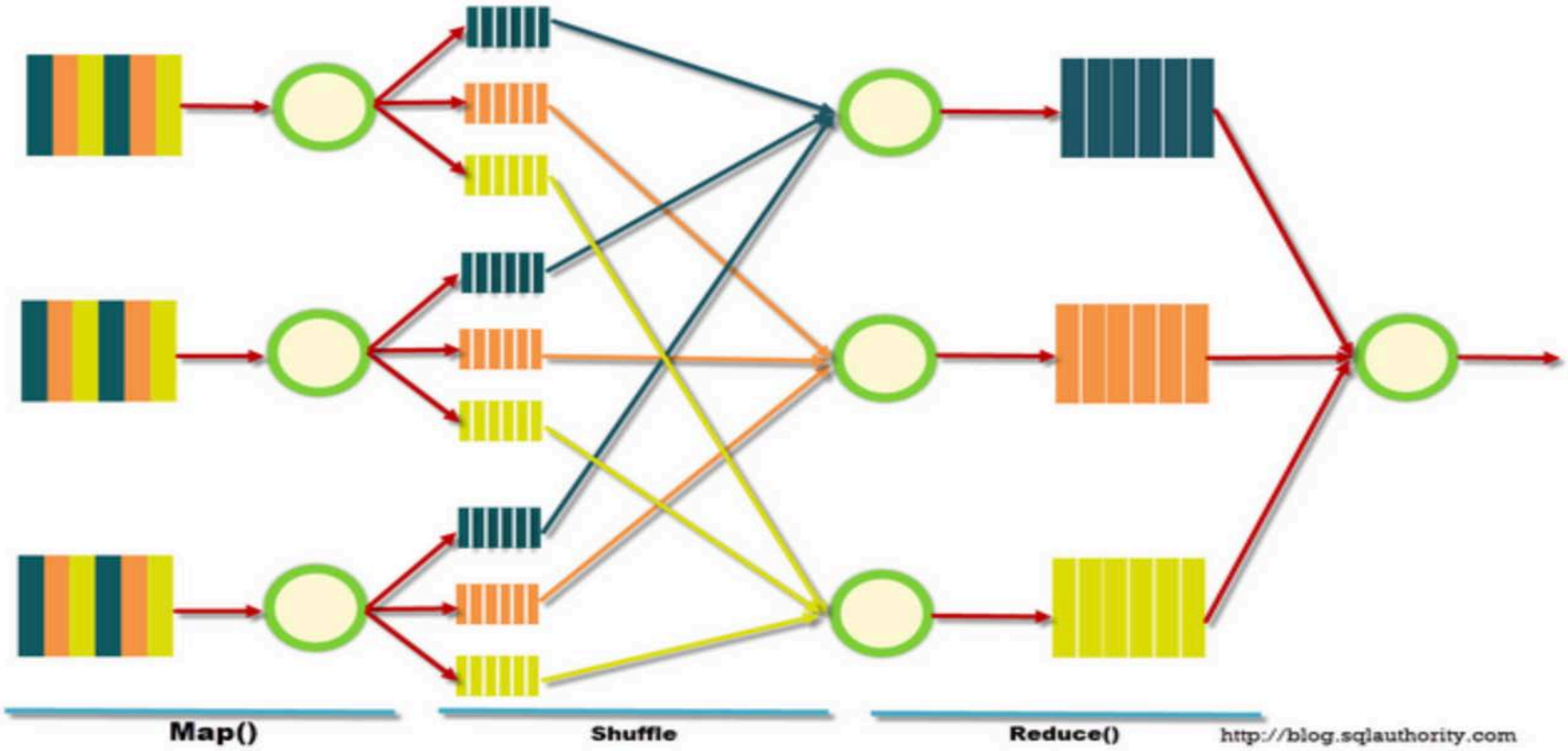


Pseudo-code: WordCount in MapReduce

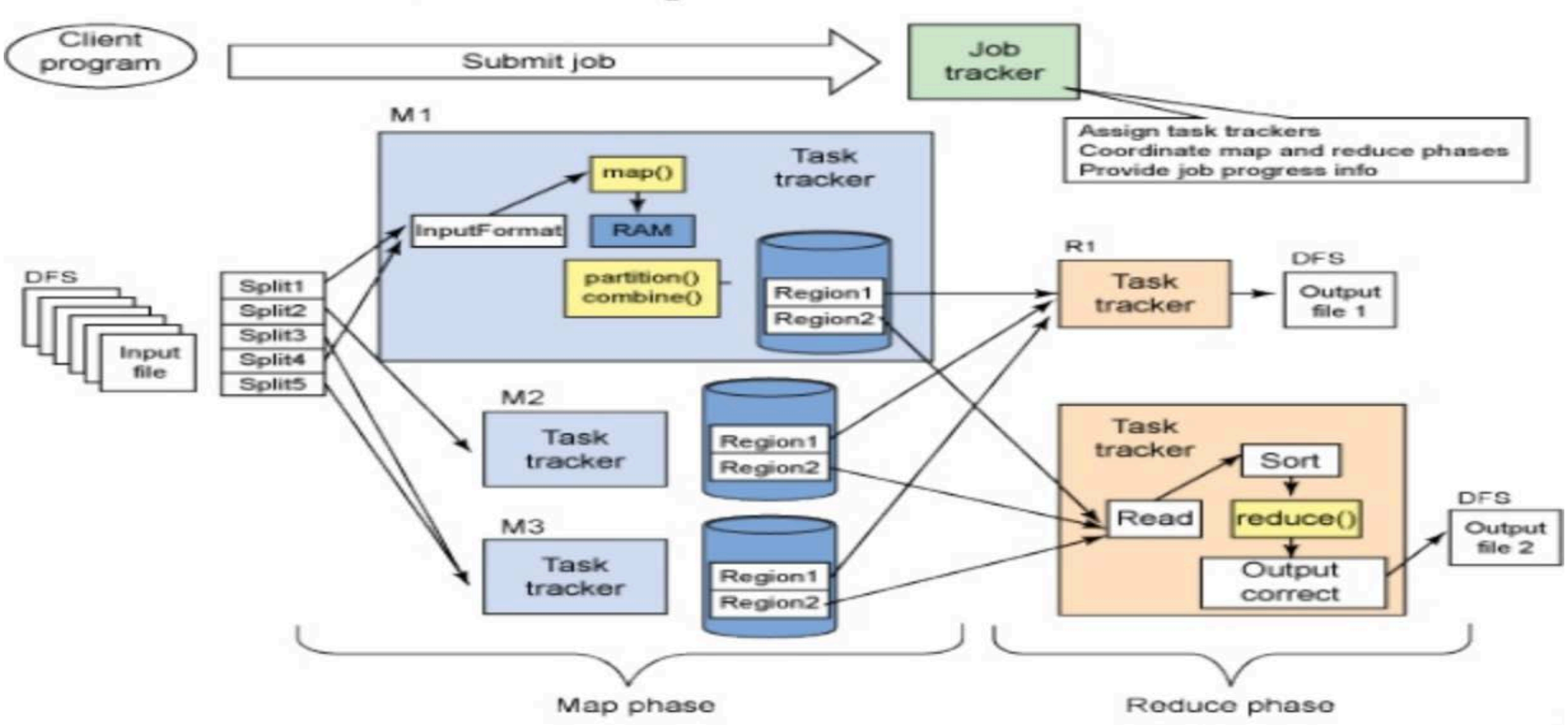
```
1: class Mapper
2:     method Map(docid a, doc d)
3:         for all term t ∈ doc d do
4:             Emit(term t, count 1)

1: class Reducer
2:     method Reduce(term t, counts [c1, c2, ...])
3:         sum ← 0
4:         for all count c ∈ counts [c1, c2, ...] do
5:             sum ← sum + c
6:         Emit(term t, count sum)
```

How MapReduce works



Architecture: Hadoop



https://commons.wikimedia.org/wiki/File:Big-data-concepts-20-638.jpg

Read a Paper: THE THREE-PASS APPROACH by S. Keshav

How to Read a Paper

S. Keshav
David R. Cheriton School of Computer Science, University of Waterloo
Waterloo, ON, Canada
keshav@ uwaterloo.ca

ABSTRACT

Researchers spend a great deal of time reading research papers. However, this skill is rarely taught, leading to much wasted effort. This article outlines a practical and efficient *three-pass method* for reading research papers. I also describe how to use this method to do a literature survey.

Categories and Subject Descriptors: A.1 [Introductory and Survey]

General Terms: Documentation.

Keywords: Paper, Reading, Hints.

1. INTRODUCTION

4. Glance over the references, mentally ticking off the ones you've already read

At the end of the first pass, you should be able to answer the *five Cs*:

1. *Category*: What type of paper is this? A measurement paper? An analysis of an existing system? A description of a research prototype?

2. *Context*: Which other papers is it related to? Which theoretical bases were used to analyze the problem?

3. *Correctness*: Do the assumptions appear to be valid?

Step 1 (I)

- Carefully read the title, abstract, and introduction
- Read the section and sub-section headings, but ignore everything else
- Read the conclusions
- Glance over the references, mentally ticking off the ones you've already read

Step 1 (II)

- **Category:** What type of paper is this? A measurement paper? An analysis of an existing system? A description of a research prototype?
- **Context:** Which other papers is it related to? Which theoretical bases were used to analyze the problem?
- **Correctness:** Do the assumptions appear to be valid?
- **Contributions:** What are the paper's main contributions?
- **Clarity:** Is the paper well written?

Step 2 (I)

- Look carefully at **the figures, diagrams and other illustrations** in the paper. Pay special attention to graphs. Are the axes properly labeled? Are results shown with error bars, so that conclusions are statistically significant? Common mistakes like these will separate rushed, shoddy work from the truly excellent

Step 2 (II)

- Remember to mark relevant unread references for further reading (this is a good way to learn more about the background of the paper)
- You can now choose to:
  - (a) set the paper aside, hoping you don't need to understand the material to be successful in your career
  - (b) return to the paper later, perhaps after reading background material or
  - (c) persevere and go on to the third pass

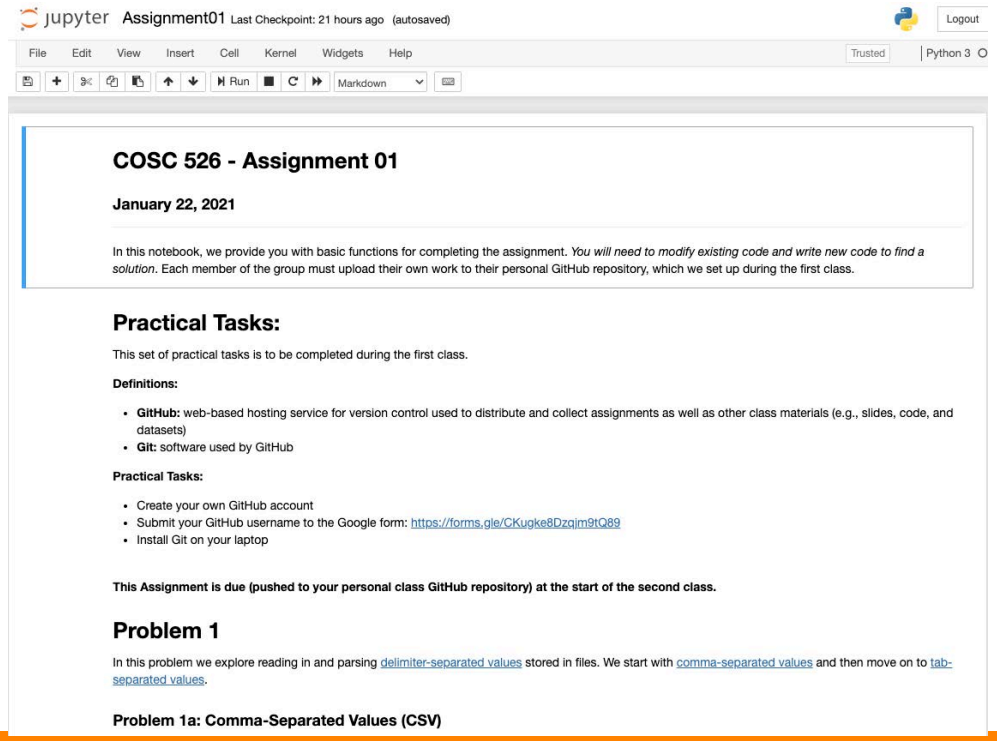
Step 3

- Attempt to virtually re-implement the paper
  - Make the same assumptions as the authors
  - Re-create the work
- Identify (and challenge) every assumption
- Think about how you yourself would present a particular idea
- Jot down ideas for future work

For next week ….

For next week (I)

- Complete and push Assignment 01 to your private repository



Free-Form Questions:

Q1. Your solutions for Problems 1 & 2 probably share a lot of code in common. You might even have copied-and-pasted from Problem 1 into Problem 2. Refactor parse_delimited_file to be useful in both problems

In [5]:
```
# Add here your code
```

Q2. Are there any pre-built Python packages that could help you solve these problems? If yes, refactor your solutions to use those packages.

In [6]:
```
# Add here your code
```

Q3. Tell us about your experience (for each point below provide a couple of sentences).

- Describe the challenges you faced in addressing these tasks and how you overcame these challenges.
- Did you work with other students on this assignment? If yes, how did you help them? How did they help you?

Write here your answers

Live Chat: The History of Big Data

Intel's Genevieve Bell shows that we have been dealing with big data for millennia, and that approaching big data problems with the right frame of reference is the key addressing many of the problems we face today from the keynote of Supercomputing 2013: https://youtu.be/CNoi-XqwJnA

List three key concepts you learned by watching the video.

```
*Write here your answers*
```


Live Chat: What we learned from 5 million books!

Live Chat:

Jean-Baptiste Michel and Erez Lieberman Aiden tell us about “What we learned from 5 million books”
https://www.ted.com/talks/jean_baptiste_michel_erez_lieberman_aiden_what_we_learned_from_5_million_books

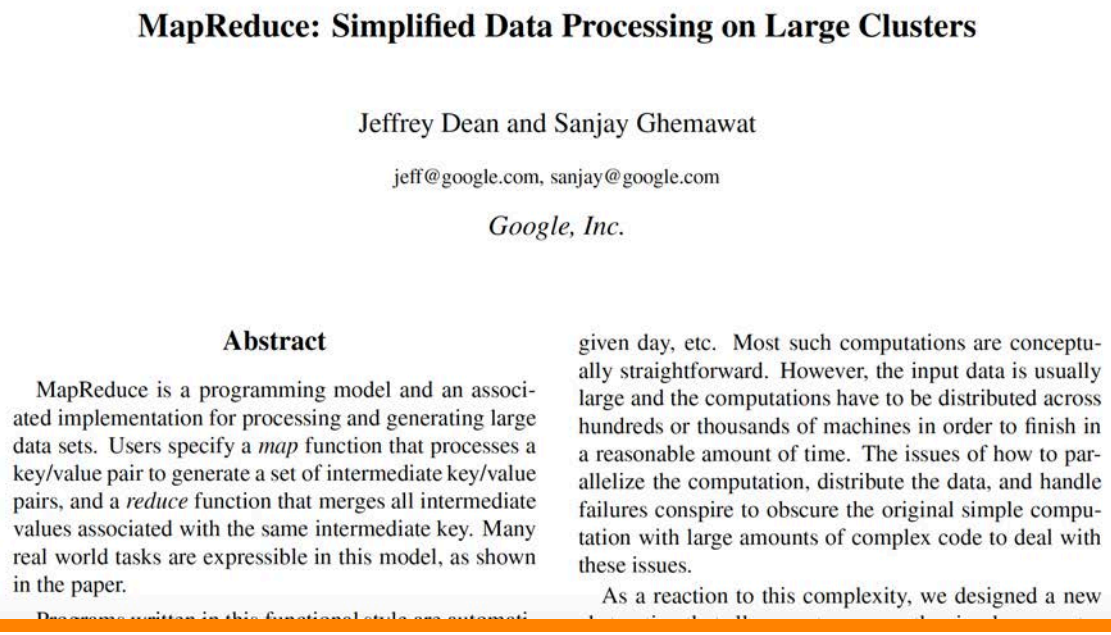
Answer these questions related to the talk:

- What is the take-away of this talk? Summarize it in up to 3 sentences.
- What are metadata?
- What is a n-gram?
- What is the suppression index?
- What is culturomics?

```
*Write here your answers*
```

Reading Assignment: MapReduce: Simpli ed Data Processing on Large Clusters

Use the three-pass approch to read the paper: Jeffrey Dean and Sanjay Ghemawat (2004) MapReduce: Simpli ed Data Processing on Large Clusters.



MapReduce: Simplified Data Processing on Large Clusters

Jeffrey Dean and Sanjay Ghemawat

jeff@google.com, sanjay@google.com

Google, Inc.

Abstract

MapReduce is a programming model and an associated implementation for processing and generating large data sets. Users specify a *map* function that processes a key/value pair to generate a set of intermediate key/value pairs, and a *reduce* function that merges all intermediate values associated with the same intermediate key. Many real world tasks are expressible in this model, as shown in the paper.

given day, etc. Most such computations are conceptually straightforward. However, the input data is usually large and the computations have to be distributed across hundreds or thousands of machines in order to finish in a reasonable amount of time. The issues of how to parallelize the computation, distribute the data, and handle failures conspire to obscure the original simple computation with large amounts of complex code to deal with these issues.

As a reaction to this complexity, we designed a new