CC5212-1
Procesamiento Masivo de Datos
Otoño 2016

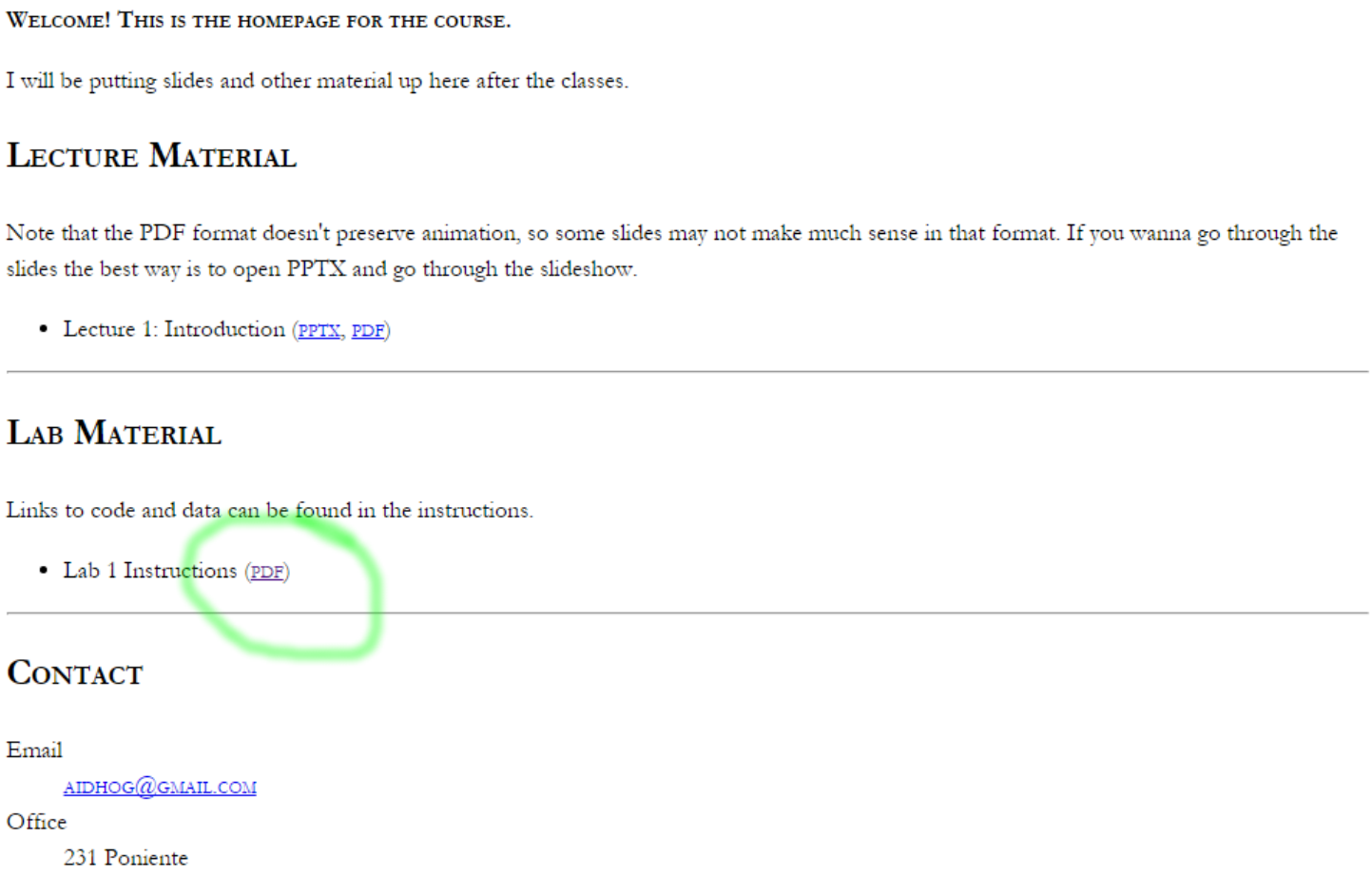
Lab 1: Wikipedia Word Count

Aidan Hogan
aidhog@gmail.com

PRÁCTICA

Instructions

- http://aidanhogan.com/teaching/cc5212-1-2016/



WELCOME! THIS IS THE HOMEPAGE FOR THE COURSE.

I will be putting slides and other material up here after the classes.

LECTURE MATERIAL

Note that the PDF format doesn't preserve animation, so some slides may not make much sense in that format. If you wanna go through the slides the best way is to open PPTX and go through the slideshow.

- Lecture 1: Introduction (PPTX, PDF)

LAB MATERIAL

Links to code and data can be found in the instructions.

- Lab 1 Instructions (PDF)

CONTACT

Email

AIDHOG@GMAIL.COM

Office

231 Poniente

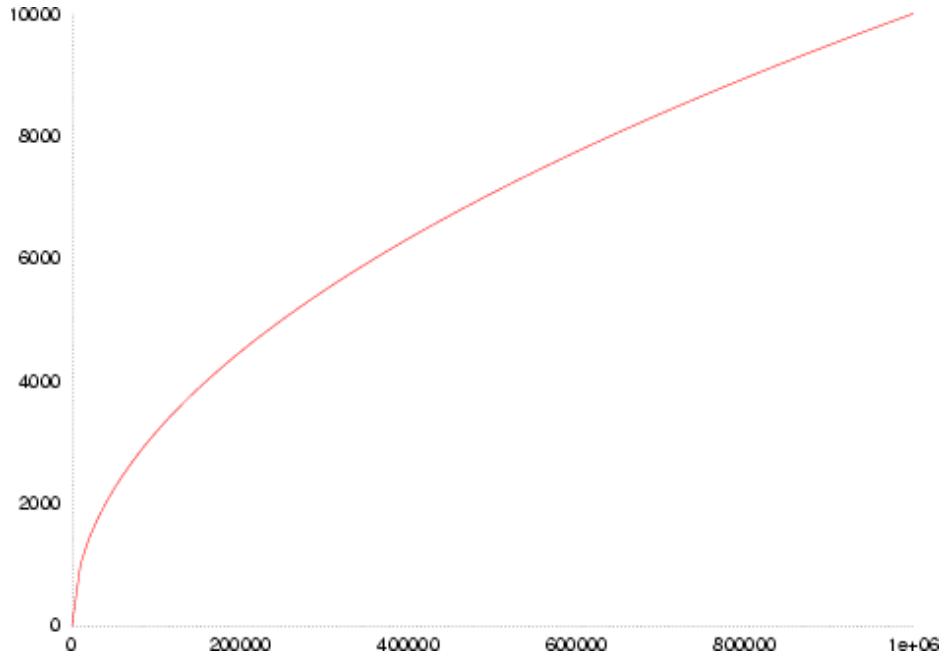
RunWordCountInMemory

- -i C:\Users\ahogan\Documents\Teaching\Data\wikipedia\es\es-wiki-abstracts.txt.gz -igz -k 100

Why did it work in memory?

We processed a lot of data. Why did it work in memory?

- Not so many unique words …
  - but lots of new proper nouns
  - Heap's law:
  - $U(n) \approx Kn^{\beta}$
  - English text
    - $K \approx 10$
    - $\beta \approx 0.6$

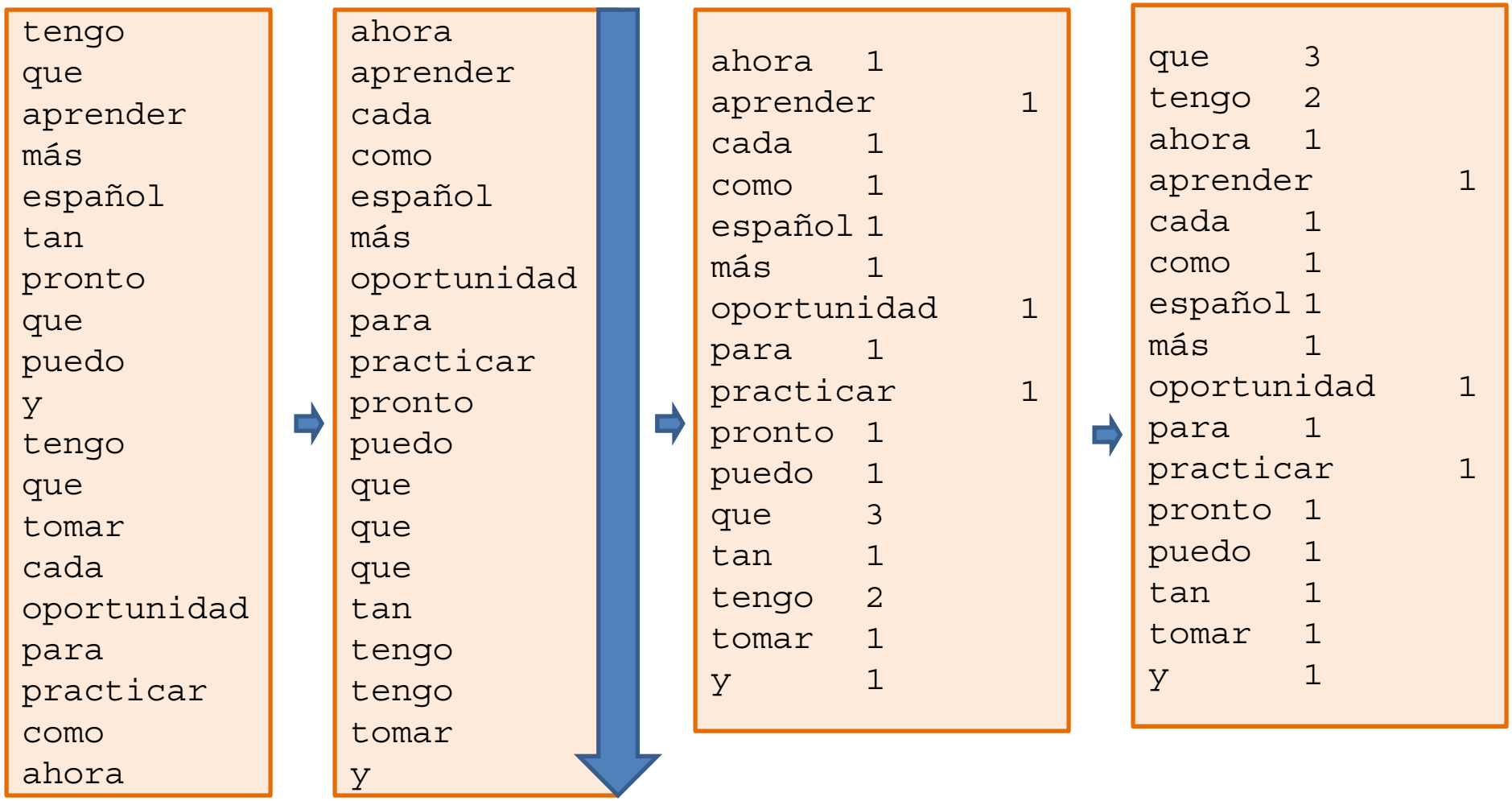


What if it doesn't work in memory?

What if it doesn't work in memory?

How could we implement a word-count (or a bi-gram count) using the hard disk for storage?

Most generic method: use sorting



How can we use the disk to sort?

-i C:\Users\ahogan\Documents\Teaching\Data\wikipedia\es\es-wiki-abstracts.txt.gz -igz -n 4

-o C:\Users\ahogan\Documents\Teaching\Data\wikipedia\es\es-wiki-abstracts-4grams.txt.gz -ogz

- -i C:\Users\ahogan\Documents\Teaching\Data\wikipedia\es\es-wiki-abstracts-4grams.txt.gz -igz -o C:\Users\ahogan\Documents\Teaching\Data\wikipedia\es\es-wiki-abstracts-4grams-s.txt.gz -ogz -tmp C:\Users\ahogan\Documents\Teaching\Data\wikipedia\es\tmp\ -b 1000000

External Merge-Sort 1: Batch

- Sort in batches

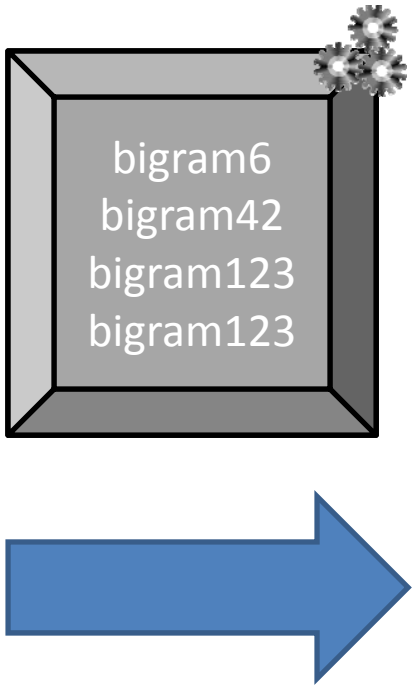
Input on-disk
(Input size: *n*)

bigram121
bigram42
bigram732
bigram42

bigram123
bigram149
bigram42
bigram1294

bigram123
bigram42
bigram6
bigram123

In-memory sort
(Batch size *b*)



Output batches on-disk
([*n*/*b*] batches)

bigram42
bigram42
bigram121
bigram732

bigram42
bigram123
bigram149
bigram1294

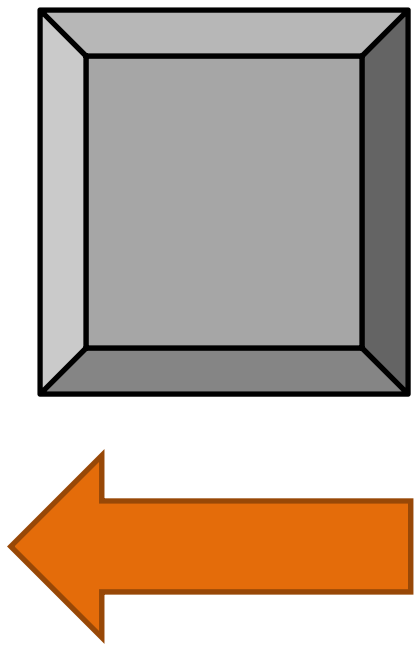
bigram6
bigram42
bigram123
bigram123

External Merge-Sort 2: Merge

Sorted output
(Output size: *n*)

bigram6
bigram42
bigram42
bigram42
bigram42
bigram121
bigram123
bigram123
bigram123
bigram149
bigram732
bigram1294

In-memory sort



Input batches on-disk
([*n*/*b*] batches)

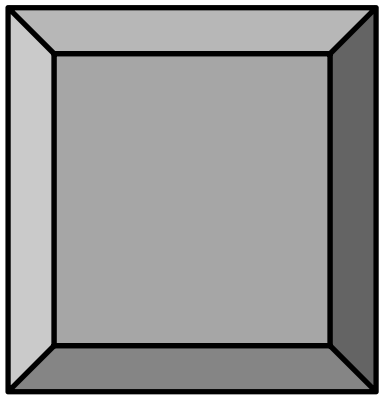
bigram42
bigram42
bigram121
bigram732

bigram42
bigram123
bigram149
bigram1294

bigram6
bigram42
bigram123
bigram123

Counting bigrams is then easy?

bigram6
bigram42
bigram42
bigram42
bigram42
bigram121
bigram123
bigram123
bigram123
bigram149
bigram732
bigram1294



Could use merge-sort again to order by occurrence!

bigram6, 1
bigram42, 4
bigram121, 1
bigram 123, 3
bigram149, 1
bigram732, 1
bigram1294, 1

Does external merge-sorting scale?

Any problem with external merge-sorting as we scale really high?

- If you have too many batches to read simultaneously, disk will go nuts

Any solution(s)?

- Use lots of main-memory to reduce batch count
- Only merge *k* at a time

If we have *n* batches and merge them *k* at a time,
how many passes will we need?

Does external merge-sorting scale?

– Use multiple machines!