Lab 1 – Wikipedia Word Count (Local)

CC5212-1

March 9, 2016

Today you are going to help me learn Spanish by finding out what are the most common Spanish words (in Wikipedia abstracts). Thanks in advance!

- First we are going to download some data:
  - `http://aidanhogan.com/teaching/data/wiki/es/es-wiki-abstracts.txt.gz`
  - `http://aidanhogan.com/teaching/data/wiki/es/es-wiki-abstracts-1k.txt`

  The first file contains the full dataset, covering all 598,385 abstracts from Spanish Wikipedia (52.5 million words, 120MB GZipped, 330MB uncompressed – no need to decompress this file, we will work with it compressed). The second file contains a sample of the first one thousand abstracts that you can open to see. We will work with the large file: we will be counting the occurrences of each unique word to find out which are the most commonly used words in Spanish (in Wikipedia abstracts at least).

- There is some code to get you started. Some parts are done, some you will have to do. Download the following file, unzip it and open it as a project in Eclipse:
  - `http://aidanhogan.com/teaching/cc5212-1-2016/lab/01/mdp-lab01.zip`

  There are some conventions we will try to stick to. To find out what the arguments refer to, simply run the class without any arguments, or with "`-h`" (so we don't have to explain the inputs here). Likewise anything you see in a command that is in grey (or Greek) here you should replace with something. Please use the time while running the classes to have a look at the code and see what it does.

- All files will be encoded in UTF-8. If you notice some encoding issues in your console, you might need to change the encoding of the console in Eclipse. See here:
  - `https://decoding.wordpress.com/2010/03/18/eclipse-how-to-change-the-console-output-encoding/`

- Please create a local blank directory to hold all your data for the lab. We will refer to this directory as PATH in future. At the end of the lab (if you are finished) you can delete this directory to save disk space. Please also try use the same directory in future labs to keep things contained and clean.

- Okay, so the first thing we want to do is to count the appearances of each unique word in the big file. The `RunWordCountInMemory` class will simply use an in-memory map to keep track of the number of times each word appears, sort the words by occurrence, and then output the top 100. We will run this over the big file. Will it work? How long will it take to count each word in all abstracts of Spanish Wikipedia? Try it and see:
  - `RunWordCountInMemory -i PATH/es-wiki-abstracts.txt.gz -igz -k 100`

  How was the runtime?

- Not challenging enough it seems? Let's try something harder: now we will count n-grams. For example a 2-gram (aka. bigram) is a sequence of two consecutive words like "`por ejemplo`", a 3-gram is a sequence of three consecutive words like "`qué raro es`". Again, there is code provided for this:

  - `RunNGramCountInMemory -i PATH/es-wiki-abstracts.txt.gz -igz -k 100 -n 2`
  - `RunNGramCountInMemory -i PATH/es-wiki-abstracts.txt.gz -igz -k 100 -n 3`
  - `RunNGramCountInMemory -i PATH/es-wiki-abstracts.txt.gz -igz -k 100 -n 4`

  Please keep going until it breaks ... or try find the largest n for which the code works.

  If you are starting to run out of memory during execution, the program will slow and eventually stop (the garbage collector gets busier and busier trying to keep the program alive). Note you can add more memory by adding, e.g., `-Xmx1024M` to the VM arguments (this gives 1024 MB of heap memory to use; you can also try more if your machine support it). Adding more memory should help you run wider n-grams until your system has no more memory to run. So add as much memory as you can and try to find the smallest n where you cannot do it in the memory available to the system.

  Now what to do? Maybe we can use the hard disk. But how?

- Let's call the smallest value of n for which the above failed "ν". First we can run the following to write the ν-grams out to a file, one ν-gram per line (all one command):

  - `ExtractNGrams -i PATH/es-wiki-abstracts.txt.gz -igz -n ν`
    `-o PATH/es-wiki-abstracts-νgrams.txt.gz -ogz`

  Now we just need to find how many times each unique ν-gram (in other words, each unique line) occurs in that file. How can we do this?

- We will sort the data, which will bunch the same lines together. Since it does not all fit in memory, we will have to sort it in memory in small *batches*, write each sorted batch to disk, and then merge all the batches. This is called an external (merge) sort. It is called external since it uses external memory (in this case, disk). It is called a merge sort since in the final stage, we will merge all the batch files. Some of the code is already provided for you, but you will have to code some tricky parts.

  - Open `ExternalMergeSort.java`
  - The `main` method is done for you. It reads input from the command line and passes them to ...
  - ... the `externalMergeSort` method, which is also done for you. It opens the input file, an output file and a temporary folder. It passes these to a method called `writeSortedBatches`, which will split the input into batches with a fixed number of lines, sort them, and write each batch to a file in the temporary folder. It then calls `mergeSortedBatches` whose job is to merge the small sorted files in the temporary folder into the output.
  - `writeSortedBatches` is already done for you, but have a look: it reads the input file (that needs to be sorted), reads `batchSize` number of lines into memory, sorts them, and writes these sorted lines to a temporary file, empties the memory buffer and continues reading from the input.
  - You need to code `mergeSortedBatches` method. This takes the list of temporary files created by `writeSortedBatches` and will merge-sort them into the output. There are some comments offering hints in the source code you downloaded. I will also offer some hints in the class.

  Once you have completed `ExternalMergeSort.java`, to sort the file output previously, run:

  - `ExternalMergeSort -i PATH/es-wiki-abstracts-νgrams.txt.gz -igz`
    `-o PATH/es-wiki-abstracts-νgrams-s.txt.gz -ogz -tmp PATH/tmp -b β`

β indicates the size of the batches: put it too high and you run out of memory, put it too low and you'll have so many batches your hard drive may explode (or just take a long time to merge the files). Try find a good value: as big as you can get it while avoiding swapping (running out of memory). Remember to give more memory using `-Xmx`.

- Now you need to count repeated lines/n-grams in the output of the previous stage. Since the file is sorted this is easy: the repeated lines will be consecutive, so we just need to scan the file and count repeated lines that appear together. Again the code is ready for you:

  – `CountDuplicates -i` PATH`/es-wiki-abstracts-`ν`grams-s.txt.gz -igz`
    `-o` PATH`/es-wiki-abstracts-`ν`grams-c.txt.gz -ogz`

  This will output a file like:

  ```
  ...
  0000000009 que rara es
  0000000007 que raro es
  0000000002 que rara esta
  0000000003 que raro este
  ...
  ```

  The lines are ordered alphabetically based on the first sort. But we would like to find the most frequent lines (those with the highest count). Note the leading zeroes on the count: we add these so we can re-use the first sorting method to do a numeric sort. If we did not have these, then "`200`" could appear before "`30`" in the ordering but after "`10`". So the zeroes allow us to sort numbers as strings.

  So we can now sort the output of the last phase using the sort, but this time we also need to add a flag to sort in descending order: we'd like the highest counts to appear first in the file. We thus need to add an '`-r`' flag; the code will do the rest. We can use the same value for β as before.

  – `ExternalMergeSort -i` PATH`/es-wiki-abstracts-`ν`grams-c.txt.gz -igz`
    `-o [path]/es-wiki-abstracts-`ν`grams-c-s.txt.gz -ogz -tmp` PATH`/tmp/ -b` β `-r`

  Now we have the top ν-grams in the data!

- But the file is really large, it might be difficult to open it to see the results. If you are on Linux, you can use `zcat [file] | more` to have a peek. If you're on Windows things are a little more difficult, maybe you can decompress the file, but otherwise there's some code included to help you. Run:

  – `Head -i` PATH`/es-wiki-abstracts-`ν`grams-c-s.txt.gz -igz -k 100`
    `-o` PATH`/es-wiki-abstracts-`ν`grams-c-s-top100.txt`

  This will give you the top 100 lines in the file you can open.

- Some questions to consider: Which of the methods above was faster and which was slower? Which of the methods above was more scalable? What were the benefits of using the hard disk vs. using main memory? Which affected the runtimes more, the size of the data or the length of the n-grams? Would it be faster to count n-grams over $2a$ abstracts, or $2n$-grams in a abstracts?

- **Submit `ExternalMergeSort.java` to u-cursos (just that file) before Monday 13th. Put the top 5 ν-grams you found in comments at the very start of the source file.**