Hands-On Data Science with the Command Line

Jason Morris - CTO
3/5/2019

nextrev
TECHNOLOGIES

Who am I?

Jason Morris, CTO: Bio & Experience

- 19+ years' experience in system architecture, research engineering, and large data analysis
- Primary focus is machine learning
  - CPU compute: TensorFlow on CUDA
  - Serverless: Lambda/Cloud Functions with Apache Spark
- Previous data science roles:
  - Big Data Engineer / Solution Architect for Databricks
  - Big Data Specialist / Certified Instructor for Amazon Web Services

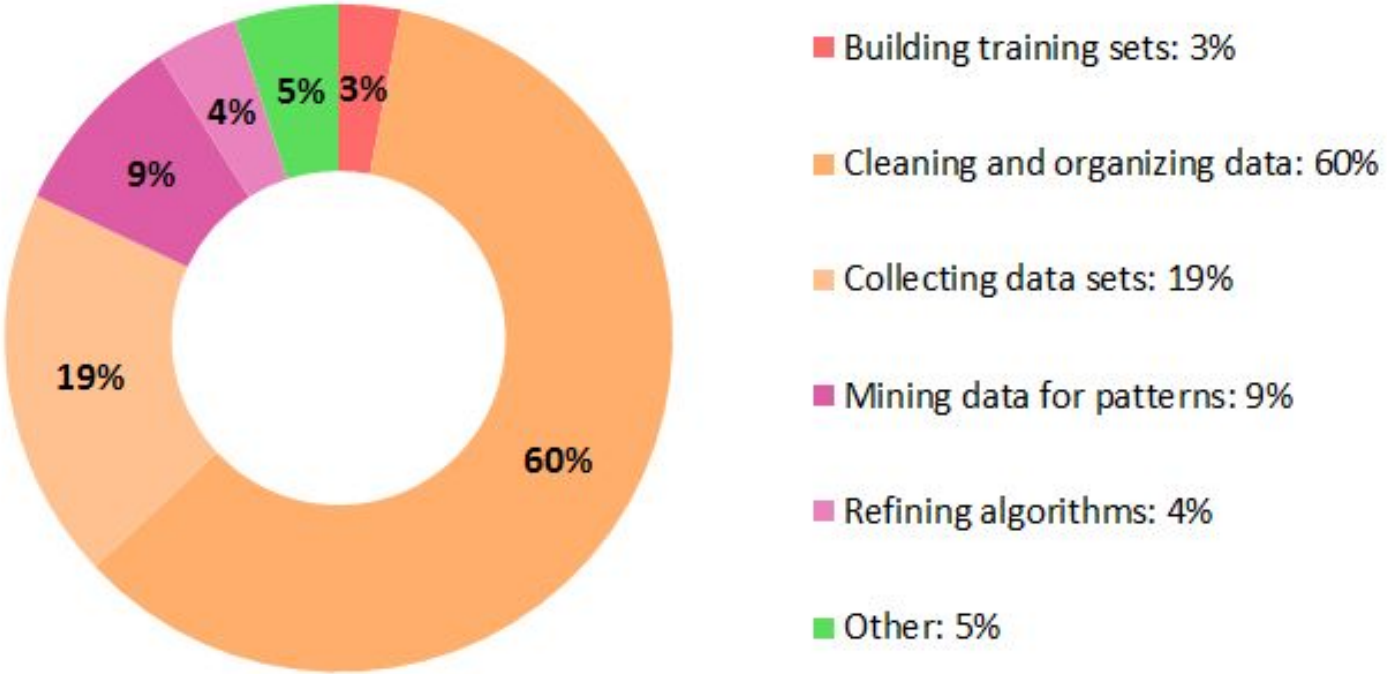
Why the Command Line?

“In the beginning...was the command line”

- The command line as been around for over 50 plus years
- Stable and extendable
- Thousands of tools pre-installed, readily available, and well documented
- One of the first things you’ll interface with on most machines
- Runs on almost everything *nix, Windows 10 +, OS X and even a toaster
- Commands executed interactively, great for experimenting and viewing results

Command line and Data Science?

Data Janitor

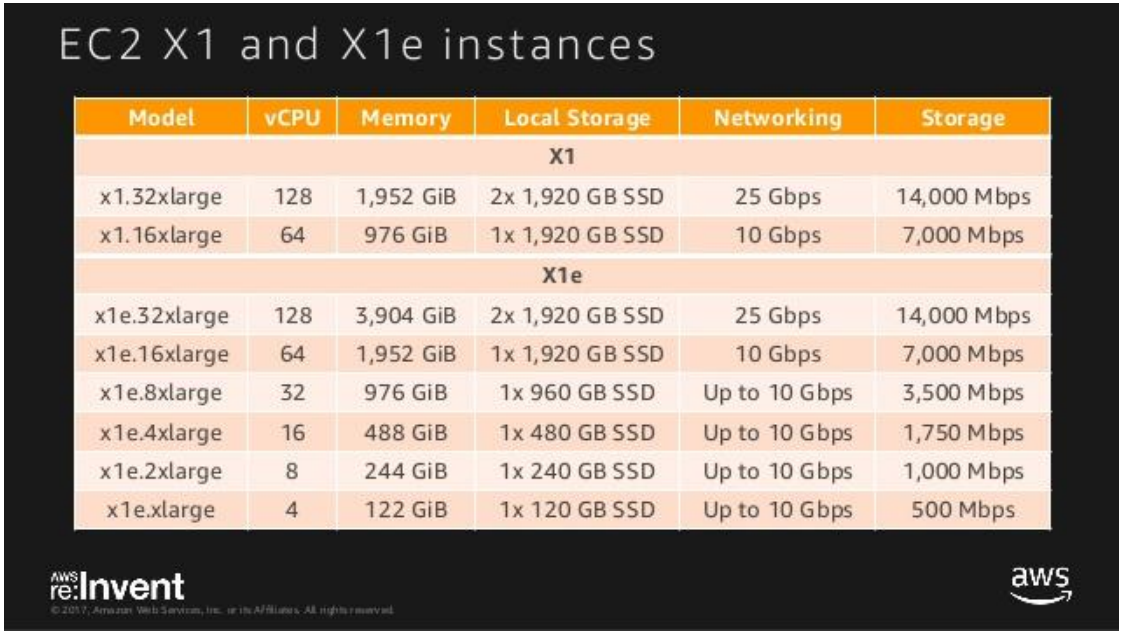


Command line and Data Science

- 60% of Data Science is cleaning and organizing data
- Tools like *sed*, *awk*, *cut*, and *tr* are still very powerful, especially for ETLing data
- For example, *tmux* is a great way to get an environment up and running to your liking. It's faster, lightweight, and very customizable

Big Data?

There's no place like 127.0.0.1 (or ::1)



Model	vCPU	Memory	Local Storage	Networking	Storage
X1					
x1.32xlarge	128	1,952 GiB	2x 1,920 GB SSD	25 Gbps	14,000 Mbps
x1.16xlarge	64	976 GiB	1x 1,920 GB SSD	10 Gbps	7,000 Mbps
X1e					
x1e.32xlarge	128	3,904 GiB	2x 1,920 GB SSD	25 Gbps	14,000 Mbps
x1e.16xlarge	64	1,952 GiB	1x 1,920 GB SSD	10 Gbps	7,000 Mbps
x1e.8xlarge	32	976 GiB	1x 960 GB SSD	Up to 10 Gbps	3,500 Mbps
x1e.4xlarge	16	488 GiB	1x 480 GB SSD	Up to 10 Gbps	1,750 Mbps
x1e.2xlarge	8	244 GiB	1x 240 GB SSD	Up to 10 Gbps	1,000 Mbps
x1e.xlarge	4	122 GiB	1x 120 GB SSD	Up to 10 Gbps	500 Mbps

Wordcount on a file?

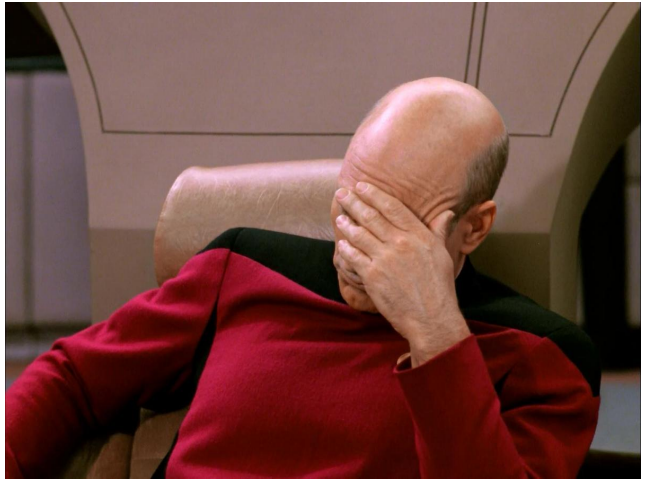
- Number of frameworks (Hadoop, Spark, pandas, etc) come to mind but everyone forgets about ~~Dre~~ the Command Line
- I asked a student how they would perform a word count on a file and they instantly replied with:
  - Using Hadoop to read the file
  - tokenize the words to form a key/value pair
  - reduce all of the keys and values that are grouped together
  - and add up the occurrences

This isn't wrong by any means and I'm not here to shame these frameworks or to tell you the command line will solve every problem on Earth

I replied “**what about using the command line?**” Their response was* :

*number 4 will shock you

I never heard of that library before, does with work with Python 3?



It was at that point I realized “someone should write a book about this….”

Hands-On Data Science with the Command Line

Automate everyday data science tasks using command-line tools

Jason Morris, Chris McCubbin
and Raymond Page

Packt
www.packt.com

What tools should I use?

If you haven’t already, feel free to follow along by cloning the repo: https://goo.gl/HjwyU1

jq - Json Processor

- Think of it like *sed* for json
- You can use it to slice and filter and map and transform structured data with the same ease that *sed*, *awk*, *grep* can do
- Perfect for working with API endpoints using *curl*

Let’s see an example:

jq in action

- GitHub has a JSON API, so let's play with that. This URL gets us the last 5 commits from the jq repo.
  - **curl 'https://api.github.com/repos/stedolan/jq/commits?per_page=5'**
- We can use jq to extract just the first commit.
  - **curl 'https://api.github.com/repos/stedolan/jq/commits?per_page=5' | jq '.[0]'**
- There's a lot of info we don't care about there, so we'll restrict it down to the most interesting fields
  - **curl 'https://api.github.com/repos/stedolan/jq/commits?per_page=5' | jq '.[0] | {message: .commit.message, name: .commit.committer.name}'**

Working with a “large” dataset

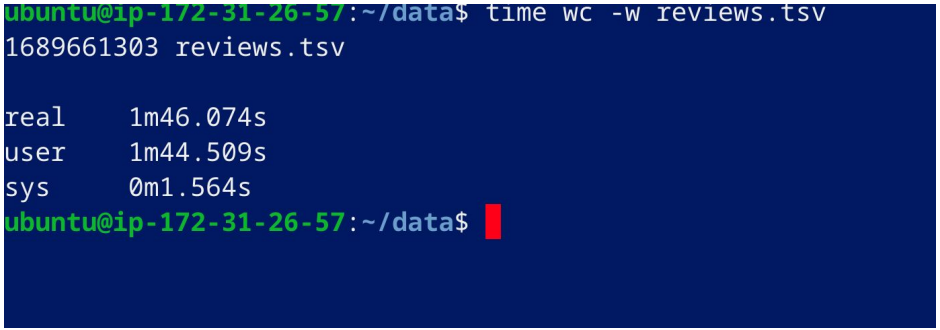
- Hopefully you downloaded this either at home are you’re using an EC2 instance (if not, no worries the data is available freely from Amazon and you can try at home)
- Let’s explore some of the everyday commands we would use to work with data such as *file*, *cat*, *head*, *more*, tail, *wc*, *cut*, and *grep*

Using *file* to determine file type

- file amazon_reviews_us_Digital_Ebook_Purchase_v1_0*
  - What was the output?
  - Do we have to decompress?
  - What does the file command actually do? (hint, man file or info file)
- Let's take a look at the first line of a data:
  - ***head* -n1 reviews.tsv**
- What's the output?

How much data are we working with?

- By performing an *ls* in our directory path, we can see we are dealing with a good amount of data
- Let's introduce wordcount (*wc)* to perform a simple wordcount with the data:
    - ***time wc* -w reviews.tsv**



Introduction to cut

- The *cut* command removes sections from each line of a file.
- Go ahead and run the following:
  - **cut -d$'\t' -f 6,8,13,14 reviews.tsv | more**
  - Let's break this down: the -d parameter tells cut we ae working with a tab separated value file, and the -f parameter tells cut what fields we are interested in
- We can go ahead and save the stripped down version of this data by doing the following:
  - **cut -d$'\t' -f 6,8,13,14 reviews.tsv > stripped_reviews.tsv**
- Let's convert to csv by doing:
  - **cat stripped_reviews.tsv | tr "\\t" "," > all_reviews.csv**

Intro to awk and grep

- *awk* is a programming language designed for text processing and typically used as a data extraction and reporting tool
- *awk* (and also *sed*) are very, very, very powerful. Both are well documented and used for a number of data parsing problems (it's the OG of ETL)
- *grep* is excellent for pattern matching. Combined with awk, you can filter out what you need
- For example, let's say we wanted to take this dataset and filter out all of the reviews that have the word "Packt" in them:
    - **cat all_reviews.csv | awk -F "," '{print $4}' | grep -i Packt**
- What else could we do with this data?

This all seems slow and sad :(

You're right

- Remember when we originally performed our word count on 'reviews.tsv'?
  - How long did it take?
  - Why do you think it was taking so long?
- Perhaps there's a way to run this wordcount in *parallel*

Introducing parallel

- *Parallel* builds and executes shell command lines from standard input in… well *parallel*!
- It can run locally or across multiple computers
- Also used as a great way to scrape websites, ingest multiple feeds of data, or ETLing a file with *sed* or *awk* in parallel

Let’s go back to our wordcount

- It took on average a minute and a half to run our wordcount without using parallel
- Let’s go ahead and run it with parallel and see our results
  - **time cat reviews.tsv | parallel --pipe --block 20M wc -w | awk '{s+=$1} END {print s}'**
  - we are ‘mapping’ a bunch of calls to wc -l , generating sub-totals, and finally adding them up with the final pipe pointing to awk the --block is the size of data we will read across each thread
- Was it faster?

Introducing sort

- *sort* sorts, merges, or compares all the lines from the given files, or standard input if none are given or for a file
- Let's go ahead and pipe one column of data (using cut) from a few lines (using head) from out dataset:
  - **zcat amazon_reviews_us_Digital_Ebook_Purchase_v1_01.tsv.gz | head -n 10 | cut -f13 | sort**
- *sort* has the ability to sort numerically as well:
  - **zcat amazon_reviews_us_Digital_Ebook_Purchase_v1_01.tsv.gz | head -n 10 | tail -n +2 | cut -f13,8 | sort -n**

Introducing sort (continued)

- Let's sort on just a part of the data (think of these as streams of data more like a database)
- We can use the -k option to sort data by columns, along with the -t option if you data is delimited by something other than tabs
    - **zcat amazon_reviews_us_Digital_Ebook_Purchase_v1_01.tsv.gz | head -n 50000 | tail -n +2 | sort -t$'\t' -k9n,9 | tail -n 1**
- Let's quickly introduce another tool called *uniq*

Introducing uniq

- *uniq* just removes adjacent identical lines in a stream of data
- In this example, we put it with sort because, usually, you can see it used with data piped from sort to count the unique values in a stream of data
    - **zcat amazon_reviews_us_Digital_Ebook_Purchase_v1_01.tsv.gz | head -n 50000 | tail -n +2 | cut -f8 | sort | uniq**
    - What is the output that was displayed?

Wrapping up

- This is only the beginning
- In my part two section (or in my book) you can learn more about visualizing the data, using the command line as a database, performing math in bash, and so much more
- Parallel is great but what about bashreduce?
- There's even a webserver called BashHTTPD that we can talk about
- And I haven't even mentioned Xonsh and Ammonite

Questions?

Thank you!

Send additional questions to jason@nextrevtech.com

More about my technical background:
https://www.linkedin.com/in/jasmor/

Spark training in May: training@nextrevtech.com (some seats left!)

We're Hiring!

nextrev
TECHNOLOGIES