the shortcut × moz://a

DATA SCIENCE

TRAINING PROGRAM

Learn effective methods to solve business problems

October 7th
@The Shortcut Lab

kodit.io selko SILO.AI

the shortcut × moz://a

DATA SCIENCE

TRAINING PROGRAM

Data Engineering

By Taneli

All the Stuff Around Data Science

Most of this course teaches how to approach data questions from a scientific point of view. This lecture and associated lab assignments will address (some of) the engineering needs around that, so that the manipulation, hypothesis testing, visualization, modeling and sometimes, actual products can actually be done.

While it is called “Engineering” as if it was a real discipline, the field is in its infancy. It often happens so that a data scientist needs to do much more engineering than actual science. And possibly so without any sort of engineering diploma.

Outline

- Job Description :)
- Big Picture Concepts
- Collection
- Storage
- Access
- Transformations
- Solutions Showcase
- Productization

Job Description :)

- In a startup or a small company
  - Data scientist or ML engineer may need to become (also) a data engineer
  - Software developer may become a data engineer
  - There might be no one to mentor
  - Solutions need to be developed with very vague requirements or specifications
  - But with a very specific time and money budget
- In an enterprise or large organization
  - Part of a (possibly larger) data team
  - Often not (directly) working on data science or implementing the production systems
  - Implementing support for data scientists
  - Supporting the production systems: data storage and access, model (re)training
- Some say, 85% of data science work is cleaning up and moving around data

Big Picture Concepts

- ETL = Extract, Transform, Load
  - Specific process
  - Resembles closely what is often wanted
- Data Warehousing
  - ETL in a business context
  - Aggregates and makes data accessible in a structured format
- Database
  - Comprehensive system for storing data
  - Database engine, API(s), query language
- Analyst systems vs production systems
  - Different SLA(s)

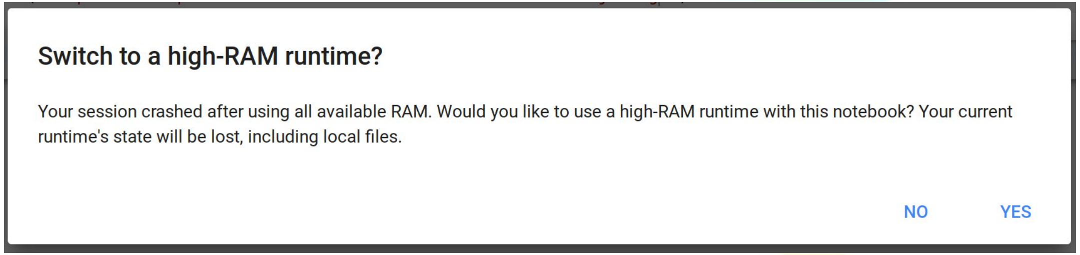
Big Picture Concepts (cont'd)

- Batch Processing and Pipeline Processing
- MapReduce
    - Distribute parallelizable computation
    - Collect results from many computers
- Apache Hadoop / Spark
    - Framework / solution to distribute data and operations
- Cloud Solutions
    - Amazon Web Services, Azure, Google Cloud Platform
    - Specify data processing and layout requirements

Motivating Example

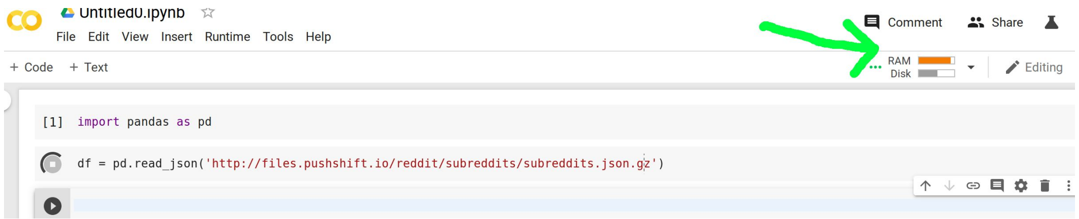
- Try this in colab:

  -
    ```
    import pandas as pd
    df = pd.read_json('http://files.pushshift.io/reddit/subreddits/subreddits.json.gz ')
    ```
  - This small 1.6 gigabyte file will fail to load even in the bigger instance (that has 25 gigabytes of RAM) → not accessible



Switch to a high-RAM runtime?

Your session crashed after using all available RAM. Would you like to use a high-RAM runtime with this notebook? Your current runtime's state will be lost, including local files.



```
[1] import pandas as pd

df = pd.read_json('http://files.pushshift.io/reddit/subreddits/subreddits.json.gz')
```

Data Engineering

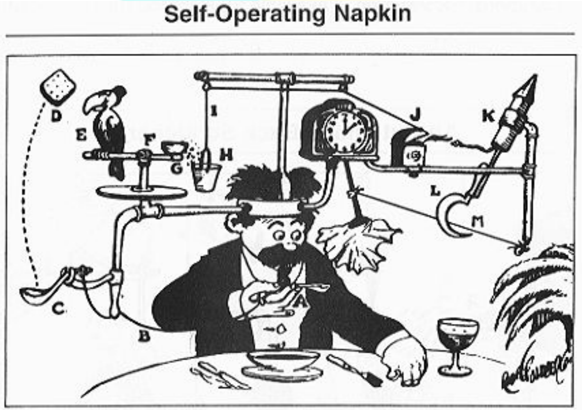
- Some questions of data engineering
  - Who are the solutions built for? How do they need to use the data?
  - How can we manage data efficiently?
    - With respect to performance of storing and retrieving
    - With respect to storage space needed
  - Can the system support growing storage needs?
    - What if the number of observations starts growing exponentially?
    - What if the number of attributes grows?
  - Does the collection support system detecting failure scenarios?
    - If no data are stored when some data is expected to have been stored
    - If access takes longer than allowed
    - If storage space is running out

Data Collection

- User input
  - Types: mouse movements, clicks, typed keys, entered text, navigation path, visit patterns
  - To support for example pirate (AARRR) metrics:
    - Acquisition, activation, retention, referral, revenue
  - Back end performance metrics
  - Challenges: GDPR, anonymization
  - A/B testing: an application of hypothesis testing

Data Collection (cont’d)

- IoT and sensors
    - Mobile phones and home appliances support collecting all kinds of sensor data
    - So do traffic lights, surveillance cameras, soil sampling equipment and smart watches
    - Sometimes specialised hardware, limited capabilities to process data (or even fix bugs)
    - Non-Python programming languages that might be in use: C, assembler, others
    - Knowledge of network protocols and their characteristics can be important
    - Data scientist may be consulted for challenges, for example
        - When capacity to send data from device to internet is limited
        - When capacity to preprocess data is limited
        - To (help) identify and remove systematic errors (caused by bugs or hardware failure)
        - To help answer hardware design questions

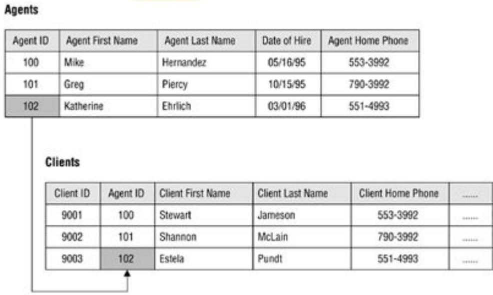

Self-Operating Napkin

Data Collection (cont’d)

- APIs
  - Programmatic way of retrieving data
  - Limitations: rate, access, legal, practical
  - In principle requests library can handle almost anything
    - Request parameters, headers, bodies
    - But need to periodically update?
    - … or provide a callback for the service to send data to?
- File Downloads
  - Surprisingly common, you already should know Kaggle
  - Typical file formats: CSV, .xlsx, .html, .zip, .txt, database dump files
  - Application specific file formats

Data Storage

- In many use cases the **raw** data is stored
  - Sometimes this is not feasible
  - This format may not be preferable for accessing it
  - If so, then transformations are needed
- Tabular Data ⇔ SQL
  - Organized into tables of rows and columns
  - Offers efficiency and versatility by separating information
  - Established standardized query language
  - There are competing solutions, with different characteristics
  - Well understood operationally
  - Distribution of data to multiple nodes implementation specific (and often quite manual)

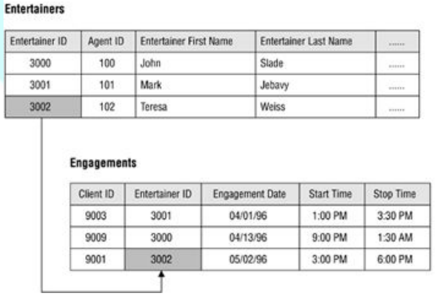


Agents

Agent ID	Agent First Name	Agent Last Name	Date of Hire	Agent Home Phone
100	Mike	Hernandez	05/16/95	553-3992
101	Greg	Piercy	10/15/95	790-3992
102	Katherine	Ehrlich	03/01/96	551-4993

Clients

Client ID	Agent ID	Client First Name	Client Last Name	Client Home Phone	....
9001	100	Stewart	Jameson	553-3992	....
9002	101	Shannon	McLain	790-3992	....
9003	102	Estela	Pundt	551-4993	....



Entertainers

Entertainer ID	Agent ID	Entertainer First Name	Entertainer Last Name	....
3000	100	John	Slade	....
3001	101	Mark	Jebavy	....
3002	102	Teresa	Weiss	....

Engagements

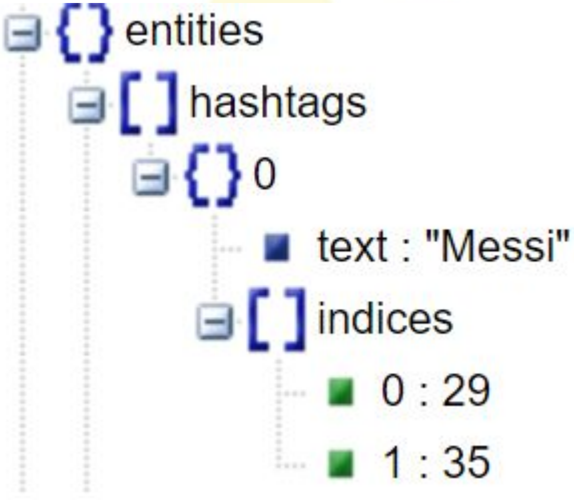
Client ID	Entertainer ID	Engagement Date	Start Time	Stop Time
9003	3001	04/01/96	1:00 PM	3:30 PM
9009	3000	04/13/96	9:00 PM	1:30 AM
9001	3002	05/02/96	3:00 PM	6:00 PM

Data Storage (cont’d)

- SQL
  - `SELECT variable1, variable2 FROM table;`
    - Include only wanted features
  - `SELECT * FROM table WHERE variable3 = 1;`
    - Include only wanted observations
  - `SELECT user.name, action.timestamp FROM user JOIN action USING (user_id);`
    - Use the same structures for data analysis and products
  - `SELECT COUNT(DISTINCT variable3) FROM table;`
    - Summary statistics directly from the database
  - `df = pd.read_sql(“SELECT * FROM table”, connection)`
    - Relatively easy access from Pandas

Data Storage (cont’d)

- Structural Data ⇔ XML, JSON, document store
  - Multiple standards and approaches to storing
  - Competing implementation strategies
    - Key/value databases
    - Document databases
    - Graph databases
  - Document stores have their own (often SQL-like) language
  - SQL engines often support storing and accessing hierarchies
- Textual data ⇔ Object storage, Fulltext database
  - Full text search databases index words after processing
  - Object storage

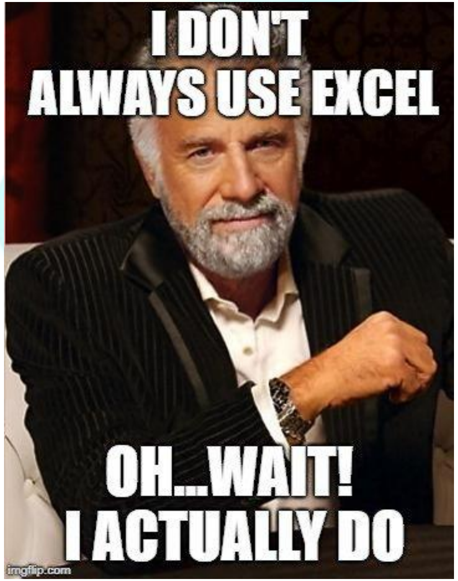


```
⊟ { } entities
    ⊟ [ ] hashtags
        ⊟ { } 0
                ■ text : "Messi"
            ⊟ [ ] indices
                    ■ 0 : 29
                    ■ 1 : 35
```

Data Storage (cont’d)

- Time Series Data
  - Specialized databases
  - For example: InfluxDb, KairosDb, Prometheus, TimescaleDb
  - Optimized for sparse nature of time observations
  - Different windowing functions

- File formats
  - CSV
  - .xlsx
  - sqlite



Data Access

- Optimizing for different access patterns
  - Are there (groups of) users, who access data very differently?
  - Can their needs be catered using one solution? Or should multiple solutions be used?

- Request processing times
  - What are the users' performance needs? What has been promised about them?
  - Measure performance using the same access methods as the users
    - Implement (or suggest to users!) more efficient methods
  - Find out the greatest bottlenecks and address them
    - Sometimes it is necessary to rethink the whole architecture
      →If all available optimizations are studied but still more than a magnitude away

Data Access (cont'd)

- Filtering
  - Random sampling
    - Some technologies, like sharding on a random hash, provide this feature for almost free
  - By observation variable (especially for wide data)
  - By time range
    - Last week of data is less than last three years
    - Can introduce bias: latest data may not be representative of all data, which can be important distinct for some applications
  - Selection based on arbitrary criteria
    - For example, observations with specific value or range is specified
    - May still be too large, or too inefficient to calculate

Data Access (cont'd)

- Large result set considerations
  - Pagination can be used when data is easy to sort (possibly by different criteria) and partial results already provide value
  - When a calculated characteristic predicts the result set size, that could be queried separately
    - Application can query the result set size first, and decide to not retrieve it if too big

- On-the-fly transformations
  - If calculation is slow, a caching solution could be feasible
  - If some transformation is always needed, consider storing it

Data Transformation

- Data access typically performed against transformed data
  - For efficiency reasons
  - For correctness reasons
  - For convenience reasons

- “Understanding” the source data
  - Parsing or converting to more usable format
  - Unifying between data sources
  - Both raw data and transformed data may be stored side-by-side
    - Can support gaining better understanding, when comparisons can be made easily

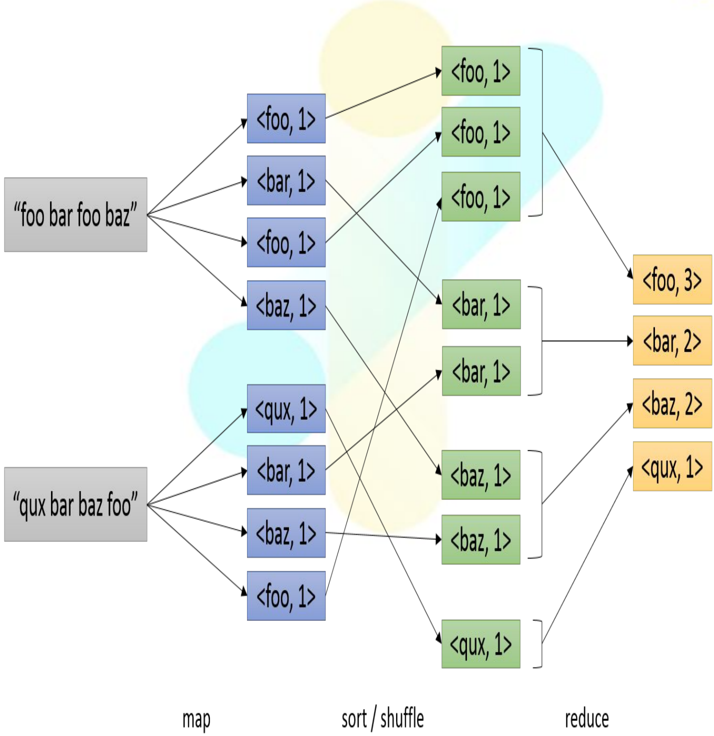
Data Transformation (cont’d)

- Erroneous data to be removed
  - Incorrectly recorded observations (which can not be recovered)
  - Sometimes analysis is needed to find them, blurring data scientist and engineer tasks
  - Masking instead of completely removing
- Collection of timing sensitive extra information
  - Some transformations have to be done in the collection phase
  - For example, what IP address did this hostname resolve to, when the request was made

INTERMISSION

MapReduce

- A MapReduce framework (or system) is usually composed of three operations (or steps):
    - Map: each worker node applies the map function to the local data, and writes the output to a temporary storage. A master node ensures that only one copy of the redundant input data is processed.
    - Shuffle: worker nodes redistribute data based on the output keys (produced by the map function), such that all data belonging to one key is located on the same worker node.
    - Reduce: worker nodes now process each group of output data, per key, in parallel.



Apache Hadoop and Spark

- Hadoop
  - Open source MapReduce (style) system
  - HDFS distributed file system to split data on multiple nodes
  - Manages the nodes, i.e. most suitable for data centers
  - Provides fault tolerance
- Spark
  - Open source streaming data processing system
  - Implements caching
  - Operates on top of Hadoop or other distributed storage
  - Can perform distributed machine learning tasks
    - Hypothesis testing, dimensionality reduction, classification, regression, clustering etc

Demo: Amazon Web Services

-

Productization

- A separate concern from improving data scientists’ access to data
- Data collection
  - Can include considerable product level work
- Implementing methods devised by data scientists **for real**
  - Making results of models available for other users
  - The application can be wildly different:
    - Guiding business strategy through internal tools
    - Providing revenue generating services
    - Providing tools for internal (or other non-paying) users
    - Even choosing which application to make

Productization (cont'd)

- Real users bring in real problems
  - Efficiency / performance: 200ms delay in model performance is still ok, 2s is not
  - 2 users will make the computers whirr, 2000 users may bring them down
  - User authentication, DoS attacks, hacking attempts, unexpected inputs
- Business intelligence
  - Often prefers 3rd party solutions over tools
  - Which may require particular layout of the data
  - Excel is BI tool par none, but not the only one
    - You may be required to choose between two tools that both cost north of 5K per month
- Data scientists
  - Care about correctness before performance
  - But when performance is an issue, the challenge may be to go from weeks to minutes

Last but not least ...

- There will be a new data set from this week on:
  - https://www.kaggle.com/dgomonov/new-york-city-airbnb-open-data