Data Acquisition

Note: This is an extended and revised set of lecture notes from DATA 301.

Overview

Data acquisition is the step of the data science process during which the data scientists determine what data is necessary to answer the questions posed to them, where this data can be found, how the data can be collected, and, following this, conduct the data acquisition/collection activities. *At the end of the data acquisition step, the data scientists are in possession of all the data that they identified for collection.*

Data acquisition involves the following:

- **[Prerequisite]** Understanding what data is.
- Knowing the **sources** of data
- Knowing the **data formats**
- **Parsing** incoming data sets into *individual values* and organizing these values into *records*.

What is Data?

In order to be able to *acquire data* we must first understand what **data** is.

***Datum*.** A **datum** is usually defined as a *single measurement* of something on a scale that is understandable to both the recorder and the reader.

Data. **Data** is the plural of ***datum***, i.e., multiple such measurements.

Data point. A **data point** is a single observation that can contain multiple individual measurements. The key point is that all measurements contribute to the observation being represented as a **data point**. Statisticians often refer to data points directly as **observations**.

Data set. A collection of **data points** of the same type, represented in a similar form (or format).

Data collection. One or more datasets combined together. Often, we abuse notation and refer to **data collections** and **data sets** (or **datasets**) simply because the key difference between the two - how many different types of data points are represented in the collection is not very relevant.

Example. Strictly speaking, a set of data points representing receipts in a store as a collection of values: Receipt number, date of receipt, total amount paid, method of payment is a **data set**. However, often we may want to augment this dataset with additional information: for example a list of items the store sells (in the form: SKU number, name of item, manufacturer, type of item, price) can be added to the list of receipts, and then, we can also store information about the individual items sold: (receipt number, List of SKUs purchased on this receipt). The three sets of data points (receipts, stock, and purchased items) technically form a data collection, however, it is often just easier to call this collection a **dataset**.

Database. A **database** is an *organized collection of data*. The key in this definition is on the work *"organized"*.

Data Engineering. A subfield of Computer Science (and, possibly, of data science) that deals with *acquisition*, *ingestion*, *transformation*, *storage*, *manipulation*, and *retrieval* of data.

In the context of the Data Science process, the following steps fall under the purview of the field of data engineering:

- Step 2: Data Acquisition
- Step 3: Data Cleaning and Integration
- Step 4: Data Modeling

In addition, data engineering is concerned with creating and maintaining the overall infrastructure for storing data, "transporting" it to the analytical methods, and archiving the results of analyses performed on the data.

Differences between Data Science and Data Engineering. As an aside, one key difference between Data Science and Data Engineering as fields of study is that of the key ways of determining success.

- **For data science** success is determined by the *accuracy of the analysis* performed and the *correctness of the predictions* made.
- **For data engineering** success is determined by the *efficiency of delivery of data* and the *efficiency of running the analytical methods*.

Data Types

Data types. Individual observations are *values* of different **types**. In Computer Science, data types are recognized as being either *atomic* or *compound*. Some data types may be treated as *atomic* in one setting and as *compound* in another.

Some *atomic data types* you will encounter are:

- Numeric data types. *Integers* and *Floating point numbers* (*reals*) are most common numeric data types. Some programming languages distinguish multiple varieties of these types based on the size (in bytes) of a single value. Integer subtypes are *byte* (1 byte), *short* (2 bytes), and *long* (4 bytes) (this assumes a regular integer is 4 bytes long). For floating point values, there is often a *double* subtype (doubles the number of bytes to store a floating point value from the regular *float*).

- Boolean data type. The Boolean data type has two values: `true` and `false`. These often are represented as `1` and `0`, and sometimes as `yes` and `no`.
- Characters. A data types that stores a single character. We owe `C` for this data type.

Some data types that can be thought of as both atomic and compound depending on the context. In data science context it is useful consider these as *atomic data types*.

- Strings. A sequence of characters is a string. It is treated as compound data type in languages that distinguish character data type from string data type. In such languages, a string is an array of characters. In data science, there is no reason to treat strings as compound values, *although there is reason to want to have access to functions* that treat strings as compound values (and access them as such).
- Date and time types. A *date*, a *time*, or a *date-time* value are compound objects by the nature of their structure: e.g., a typical date consists of a *day*, *month* and *year*, and a typical *time* value may have an *hour* component, a *minute* component and a *second* component. Again, in data science, these values show up often, and it is convenient to treat them as *atomic* values for the purposes of representation, but *compound* values for the purposes of access. Being able to extract the individual components from these values is an important feature to have.

Some of the compound data types we will see in this course:

- Arrays. An array is a collection of values of the same *base type* of a known length, in which, each base type value is associated with a position.
- Lists. A list is a collection of values (typically of the same *base type*), in which for each value, the *next value* is known. Lists are not bound by a specific pre-defined length, and (if treated as mutable) can be of different lengths throughout their lifetime.
- Sets and Bags. A set is a collection of values where each value can occur at most once (i.e., given a value of a base type it is either *in* a given set, or it is *not in* a given set). A bag is a collection of values where each value can occur multiple times. Sometimes, bags are represented as functions: given a value, the bag function returns the number of time the value is found in the bag.
- Dictionaries. A dictionary is a collection of key-value pairs. A **key-value pair** is a pair of values x:y where x is usually a string called key representing the ”name” of the value, and y is a value of any type that is said to correspond to the key. A dictionary is then, a collection of key-value pairs, where all keys are **unique**.
- Records. A single record is essentially a dictionary with a pre-defined set of keys.

Data Representations and Data Formats

An individual data point can be as simple as a single atomic value (e.g., a number) and as complex as, for example, a dictionary of key-value pairs, where some of the keys have values that are lists of other dictionaries.

Data Representation is the means of rendering the *data type* of a data point in a way that is readable by humans and/or software.

Data Formats are certain types of data representations often used for storing data points in files.

Data representation refers to the semantics, i.e., the meaning of data. Data format refers to the syntax, i.e., the means of storing data in files.

Types of Data Representations (and their Data Formats)

There is a number of archetypes of data representations that are often used in data science and data engineering.

A list of most popular such (arche)types with short explanations is shown below. Individual data representations are discussed in more detail in separate lecture notes.

Data representations.

- **Tabular Data.** Represents a dataset is a two-dimensional table, where each *row* represents a single data point, and each *column* represents one type of observation (a specific *datum*) from that data point. Traditionally, individual column values are expected to have atomic data types. Typical data formats: *Comma-Separated Values files (CSV files)*, *Tab-Separated Values files (TSV files)*, *spreadsheets*, *relational databases*.
- **Structured Data.** Essentially a data representation, where each data point is presented in a form of a (possibly complex and multi-tiered) dictionary. The keys correspond to the individual observations in the data point, the values may be of atomic, or compound types. All data points are represented using the same collection of keys. Typical data formats: *JSON*, *XML*.
- **Semistructured Data.** Semistructured data representation diverges from structured data in two major ways. First, the condition that all data points in a dataset are represented using the same collection of keys is omitted, and as such, *some* semistructured data representations, as essentially representations of datasets where each data point warrants a dictionary with its own set of key-value pairs (and its own relationship between keys and data types of the values). Additionally, some notions of semistructured data involve representing *portions of data points* outside of the key-value pair structure. Typical formats for semistructured data are XML, JSON and HTML.
- **Textual Data.** Data representation for cases where a single data point in a dataset is a *long string* (i.e., a single text document). Textual data includes a number of implicit ways of breaking a single data point (a single text document) into sub-parts: words, sentences, paragraphs, chapters, and so on. Typical data formats: *Text (.txt) files*, *MS Word documents*, *PDF documents*, *HTML files*, *SGML files*.

In addition to these ways of representing specific data sets, there are two more *cross-cutting concerns* related to data representation, that may be relevant to this course.

- **Temporal data.** A dataset is said to contain temporal data if one or more observations in each data point are of type date/time, and represent (to a certain level of granularity) a specific moment in time, when the observation represented by the data point was made.
- **Geographic** or **geolocation data.** A dataset is said to contain geographic or geolocation data if one or more observations (or a group of observations combined) in each data point contain values that are sufficiently precise to determine a specific geographic location of the observation. Typically,

this means the presence of the *longitude* and *lattitude* coordinates of a specfic location (when referred to Earth), but sometimes, other coordinate systems can be used (e.g., to specify a location in space, relative to Earth).

Data Acquisition as a Process

Things to understand about the data acquisiton process:

Sources of data. Data needed for the analysis can come from a variety of sources.

- Internal sources. Some datasets may already be collected and available as part of the overall data collection of the organization. This is true both for *business-centric* data science and *science-centric* data science.
  - **For business-centric** data science, a lot of the data is available in the transaction data bases that the organization uses to record its day to day operations (personnel/HR, clients and suppliers, sales, branches, and so on).
  - **For science-centric** data science, by the time the data science process starts in earnest, there is typically a collection of scientific observations that already has been built.
- Existing external sources. Some necessary data may be available in ready form from an outside source (either for free, or for a fee).
- External sources requiring collection efforts. Some data may be available from an outside source or sources, but acquisition of data may require special-purpose processing.

Example. An on-line retailer wants to see if there is a relationship between the sales revenue on a given day (or a given week) and two other factors: the Dow Jones Industrial Index (which the on-line retailer interprets as a proxy for ”how economy is doing at the moment”), and the volume and the sentiment of mentions of the on-line retailer on popular social networks and streaming media (Facebook, Twitter, Pinterest, Yelp).

To answer this question, the data science team needs to acquire the following data:

1. **Sales revenue numbers by date.** This data is available **internally**. The on-line retailer is maintaining a sales database that stores information about the date every single purchase and the amount paid to the retailer on every single receipt. Depending on other factors, the actual *aggregated data* may be already available to the data science team (if the data was aggregated by day for a report at an earlier time), or the data science team may need to extract the appropriate information from the sales database as part of the acquisition process.
2. **Dow Jones Industrial Index tracking data.** This data is readily available from one or more outside vendors, and thus is classified as an **existing external source.** The on-line retailer pays one of the vendors to obtain the data.
3. **Number of mentions of the on-line retailer (and the associated sentiment) on Facebook, Twitter, Pinterest and Yelp**. This data is **available in parts from the four external sources**: Facebook, Twitter, Pinterest and Yelp. Each source provides either an API (for free or for a fee) that allows for collection of information relevant to the mentions of the on-line retailer, or the information can

be obtained by *scraping*: i.e., visting certain parts of the websites controlled by the specific sources, and collecting the information form those parts. The team of data scientists must create a collection of scripts that obtain the information from these four sources. For Facebook and Twitter the scripts use the provided API and request all information about new mentions of the on-line retailer's name. For Pinterest and Yelp, the scripts download the on-line retailer's "landing pages" on each of the sites once a day, and collect and new content avaialble from these pages.

As such, in order to acquire the data necessary to answer the posed questions, the data science team needs to obtain datasets from all three of the abovementioned types of sources.

Data acquisition process. To obtain a specific dataset, the data science team may need to perform one or more of the following actions.

- **Nothing.** The dataset may already exist in exactly the same form as needed, and the data science team may already have possession of it. This is often true for the datasets coming from *internal sources*.
- **Identify the source and request/download.** Readily available datasets that are not in the data science team's possession essentially require just a download, or simple transfer of data. This is true both for internal and for some external datasets (e.g., any data from the US Census Bureau is ready for download, the key is identifying what specifically is needed).
- **Develop a set of data extraction queries/tools.** Some data is needed in a form that aggregates or otherwise transforms the information stored in an existing data collection/database. To build an appropraite dataset, the data science team must prepare a set of data extraction tools, or, if the data is stored in a relational database (as is often the case), a set of data extraction queries.
- **Implement data extraction infrastructure.** Commonly used for outside sources that require extra collection efforts, implementation of data extraction infrastructure involves building and deploying software for collecting individual data observations from the outside sources, as these observations become available.
- **Implement a brand new data capture infrastructure.** Sometimes, to answer the posed questions, the data science team needs to acquire the data that does not exist - i.e., the data science team needs to *originate a specific set of measurements and observations*. In this case, the data science team may need to instrument the entire data capture infrastructure, possibly from the sensors capturing the observations in the physical world (if needed), all the way to the data extraction and data preparation/formatting software.

Data formats. When the data science team needs multiple datasets for future analysis, the data may come in a variety of representations and formats. In such cases, some observations need to be made:

- **This may not be the right stage** to reconcile all the data. Data cleaning and data modeling steps may be more appropriate for placing the data into the form that will be needed for the analysis.
- **Take note of the data formats** in which the data comes in. While now may not be the time to do anything about different data formats, knowing how each dataset is represented will help on followup steps.

- **Select compatible data formats** if the data representation depends on the data science team.

  For example, if multiple scripts are built to collect similar data from different sources, storing all results in similarly structured tabular representations is a good idea.

- **Select the simplest data representations and data formats** possible. If data is well suited for tabular representation, no need to build complex JSON objects out of it in many cases.

Data Velocity

One of the core aspects influencing the specifics of *data acquisition* in a specific project is the **velocity of data** needed to be collected.

In Data Science the term Data Velocity refers to the *changes in the volume of to be processed data over time*.

While Data Velocity is essentially a continuum, it is convenient to split the data into the following categories (sorted from lowest to highest velocity):

1. **Static.** Static data is represented by a single (however large!) dataset that needs to be processed exactly once.
2. **Batch.** Batch data is data that remains static for fairly large periods of time, but does have occasional *large* updates. These updates trigger reruns of the analysis.
3. **Periodic.** Periodic datasets come with frequent updates, however the time to process the updates is significantly shorter than the time between the consecutive updates. The updates are incorporated into on-going analysis.
4. **Near real-time.** Near real-time data velocity occurs when the data is *streaming*, but its processing can be done with some pooling and delay. The updates are immediately incorporated into on-goinging analysis.
5. **Real time.** Data is *streaming* and immediate reaction is required. Analytics is not longer done in batches.

Acquisition of Static Datasets. A fairly straightforward process.

1. Obtain the actual copy of the data, host it locally.
2. Develop data parsers necessary to process the dataset into initial data model.
3. Run the parsers on the data set, obtain the model.

Processing time is not a big issue, as this process runs only once per analysis.

Acquisition of Batch Datasets. Short term, same strategies as for Static Datasets can be employed. Medium-to-long term, a choice between the two options can be made:

1. **Automate** as much of the process for data procurement. Deploy the automated software for data procurement as part of the overall data science analytical software pipeline.

2. Keep the data procurement **manual**, i.e., when updates become available, manually procure and deploy them and manually trigger the rerun of the analytical pipeline.

Processing time is not a big issue again, as the time between consecutive updates is significantly longer than update processing.

Automation requires resources, and may be poor ROI if data updates are rare and their procurement/deployment is a simple procedure.

Acquisition of Periodic Datasets. This is where acquisition strategy starts changing drastically.

- It is typically advisable **to automate** as much processing of periodic datasets as possible.
- If impossible, a repeatable manual procedure needs to be developed, and it needs to become a designated task for someone on the data science team.
- Processing time starts mattering a bit.

Acquisition of Near Real-Time Datasets. Manual operations are no longer attainable. There are two possible scenarios for near real-time data procurement:

- *Data pooling at the source.* This is essentially a scenario where the data source acts exactly like in the case of batch data, only with increased update frequency. That is, new data is pooled/buffered by the data source and is released in short bursts.
- *Data is streaming but can be pooled in the data acquisition process.* In some cases, the data source pushes individual/atomic data updates as soon as they come (streams the data), however the nature of the data allows for pooling the individual updates, and processing of a pooled collection of data as it accumulates over a short period of time.

In either case:

- Acquisition process must be **automated**.
- Processing time **matters a lot**.

Acquisition of Near Real-Time Datasets. In this case, data acquisition is part of uninterrupted chain of software that is ”always up”.

- All data is streaming, i.e., the processing pipeline is receiving the data as it is issued by the data source.
- All processing must be automated. The data items need to be immediately passed over up the data science pipeline for proper modeling and analysis.
- Processing time matters a lot.
- Core concern is **data overload**: the situation where processing of a single icoming data item takes longer than the average time between incoming data items.
- Infrastructures may need to be built to protect from data overload.

Data Velocity: General Observations

Batch size. The size of the data that needs to be processed is usually in inverse proportion with update frequency. There are usually **one-to-three orders of magnitude difference** between the sizes of processed as one moves from real-time to near-real time to pereodic to batch to static data.

Processing techniques. Very large datasets can be efficiently processed with the help of **distributed computing architectures** such as Apache Hadoop and Apache Spark. These architectures best fit static and batch data, and will work well with pereodic data.

On the other end of the spectrum, real-time and near real-time data processing requires fast and agile processing of individual data items. Typically, massively distributed *eventually consistent DBMS* or distributed RDBMS are used.

A short summary of techniques is shown in the table below:

Special Case: Data Creation

Question: What is the data needed for analysis does not exist?

In such cases, the Data Science team may need to **create new data from scratch**.

Physical measurements. In rare cases, the creation of new data from scratch may involve building a brand new infrastructure for collecting new, previously unavailable measurements.

Opinion discovery. Often, the missing data can be produced through traditional survey research means: the data science team develops appropriate survey instrumentation, deploys it and collects the data.