Assess Superstep

The objectives of superstep are to show you how to assess your data science data for invalid or erroneous data values. I urge that you spend the time to "clean up" the data before you progress to the data science, as the incorrect data entries will cause a major impact on the later steps in the process. Perform a data science project on "erroneous" data also, to understand why the upstream processes are producing erroneous data. I have uncovered numerous unknown problems in my customers' ecosystems by investigation.

Data quality refers to the condition of a set of qualitative or quantitative variables. Data quality is a multidimensional measurement of the acceptability of specific data sets. In business, data quality is measured to determine whether data can be used as a basis for reliable intelligence extraction for supporting organizational decisions.

Data profiling involves observing in your data sources all the viewpoints that the information offers. The main goal is to determine if individual viewpoints are accurate and complete. The Assess superstep determines what additional processing to apply to the entries that are noncompliant. Minor pieces of incorrect data can have major impacts in later data processing steps and can impact the quality of the data science.

1- Errors

Did you find errors or issues? Typically, I can do one of four things to the data.

- **Accept the Error**

  If it falls within an acceptable standard (i.e., West Street instead of West St.), I can decide to accept it and move on to the next data entry.

  Take note that if you accept the error, you will affect data science techniques and algorithms that perform classification, such as binning, regression, clustering, and decision trees, because these processes assume that the values in this example are not the same. This option is the easy option, but not always the best option.

- **Reject the Error**

  Occasionally, predominantly with first-time data imports, the information is so severely damaged that it is better to simply delete the data entry methodically and not try to correct it. Take note: Removing data is a last resort. I normally add a quality flag and use this flag to avoid this erroneous data being used in data science techniques and algorithms that it will negatively affect. I will discuss specific data science techniques and algorithms in the rest of this book, and at each stage, I will explain how to deal with erroneous data.

- **Correct the Error**

  This is the option that a major part of the assess step is dedicated to. Spelling mistakes in customer names, addresses, and locations are a common source of errors, which are methodically corrected. If there are variations on a name, I recommend that you set one data source as the "master" and keep the data consolidated and correct across all the databases using that master as your primary source. I also suggest that you store the original error in a separate value, as it is useful for discovering patterns for data sources that consistently produce errors.

- **Create a Default Value**

  This is an option that I commonly see used in my consulting work with companies. Most system developers assume that if the business doesn't enter the value, they should enter a default value. Common values that I have seen are “unknown” or “n/a.” Unfortunately, I have also seen many undesirable choices, such as birthdays for dates or pets' names for first name and last name, parents' addresses . . . This address choice goes awry, of course, when more than 300 marketing letters with sample products are sent to parents' addresses by several companies that are using the same service to distribute their marketing work. I suggest that you discuss default values with your customer in detail and agree on an official “missing data” value.

2-Analysis of Data

I always generate a health report for all data imports. I suggest the following six data quality dimensions.

- **Completeness**

  I calculate the number of incorrect entries on each data source's fields as a percentage of the total data. If the data source holds specific importance because of critical data (customer names, phone numbers, e-mail addresses, etc.), I start the analysis of these first, to ensure that the data source is fit to progress to the next phase of analysis for completeness on noncritical data. For example, for personal data to be unique, you need, as a minimum, a first name, last name, and date of birth. If any of this information is not part of the data, it is an incomplete personal data entry. Note that completeness is specific to the business area of the data you are processing.

- **Uniqueness**

  I evaluate how unique the specific value is, in comparison to the rest of the data in that field. Also, test the value against other known sources of the same data sets. The last test for uniqueness is to show where the same field is in many data sources. You will report the uniqueness normally, as a histogram across all unique values in each data source.

- **Timeliness**

  Record the impact of the date and time on the data source. Are there periods of stability or instability? This check is useful when scheduling extracts from source systems. I have seen countless month-end snapshot extracts performed before the month-end completed. These extracts are of no value. I suggest you work closely with your customer's operational people, to ensure that your data extracts are performed at the correct point in the business cycle.

- **Validity**

  Validity is tested against known and approved standards. It is recorded as a percentage of nonconformance against the standard. I have found that most data entries are covered by a standard. For example, country code uses ISO 3166-1; currencies use ISO 4217.

  I also suggest that you look at customer-specific standards, for example,

  International Classification of Diseases (ICD) standards ICD-10. Take note: Standards change over time. For example, ICD-10 is the tenth version of the standard. ICD-7 took effect in 1958, ICD-8A

in 1968, ICD-9 in 1979, and ICD-10 in 1999. So, when you validate data, make sure that you apply the correct standard on the correct data period.

- **Accuracy**

Accuracy is a measure of the data against the real-world person or object that is recorded in the data source. There are regulations, such as the European Union's General Data Protection Regulation (GDPR), that require data to be compliant for accuracy.

I recommend that you investigate what standards and regulations you must comply with for accuracy.

- **Consistency**

This measure is recorded as the shift in the patterns in the data. Measure how data changes load after load. I suggest that you measure patterns and checksums for data sources.

Why Missing Value Treatment Is Required
Explain with notes on the data traceability matrix why there is missing data in the data lake.
Why Data Has Missing Values

The following are common reasons for missing data:
- Data fields renamed during upgrades
- Migration processes from old systems to new systems where mappings were incomplete
- Incorrect tables supplied in loading specifications by subject-matter expert
- Data simply not recorded, as it was not available
- Legal reasons, owing to data protection legislation
- Someone else's "bad" data science. People and projects make mistakes, and you will have to fix their errors in your own data science.

Practical Actions for Missing Values
The Python package pandas enables several automatic error-management features.
Following are four basic processing concepts

 Drop the Columns Where All Elements Are Missing Values
```
TestData=RawData.dropna(axis=1, how='all')
```

 Drop the Columns Where Any of the Elements Is Missing Values
```
TestData=RawData.dropna(axis=1, how='any')
```

 Keep Only the Rows That Contain a Maximum of Two Missing Values
```
TestData=RawData.dropna(thresh=2)
```

 Fill All Missing Values with the Mean, Median, Mode, Minimum, and Maximum of the

Particular Numeric Column

```
TestData=RawData.fillna(RawData.mean())
TestData=RawData.fillna(RawData.median())
TestData=RawData.fillna(RawData.mode())
TestData=RawData.fillna(RawData.min())
TestData=RawData.fillna(RawData.max())
```

Graph Theory

I provide a simple introduction to graph theory, before we progress to the examples. Graphs are useful for indicating relationships between entities in the real world. The basic building blocks are two distinct graph components, as follows

Node

The node is any specific single entity. For example, in “Andre kiss Zhaan," there are two nodes: Andre and Zhaan (Figure 8-1).

Figure 8-1. *Nodes Andre and Zhaan*

Edge

The edge is any specific relationship between two nodes. For example, in “Andre kiss Zhaan," there are two nodes, i.e., Andre and Zhaan. The edge is “kiss." The edge can be recorded as non-directed. This will record “kiss” as “kiss each other” (Figure 8-2).

Figure 8-2. *Nodes Andre and Zhaan kiss each other*

The edge can be recorded as directed. This will record the “kiss” as “kiss toward” (Figure 8-3).

Figure 8-3. Nodes Andre kiss toward Zhaan

This enforces the direction of the edge. This concept of direction is useful when dealing with actions that must occur in a specific order, or when you have to follow a specific route. Figure 8-4 shows an example of this.

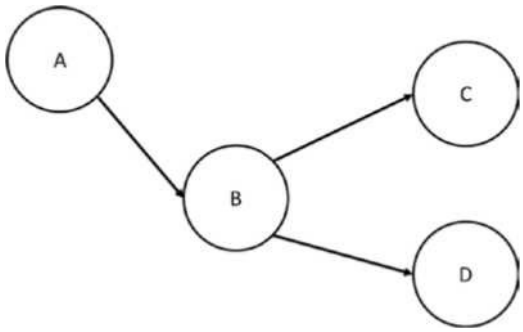

Figure 8-4

Directed Acyclic Graph (DAG)

A directed acyclic graph is a specific graph that only has one path through the graph. Figure 8-5 shows a graph that is a DAG, and Figure 8-6 shows a graph that is not a DAG.

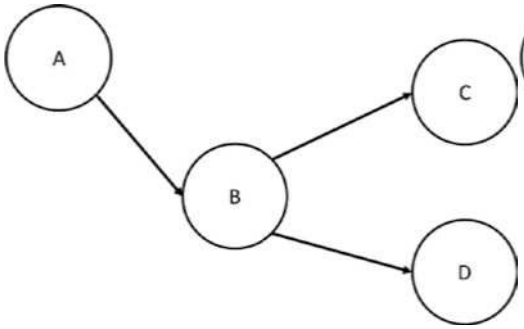

Figure 8-5

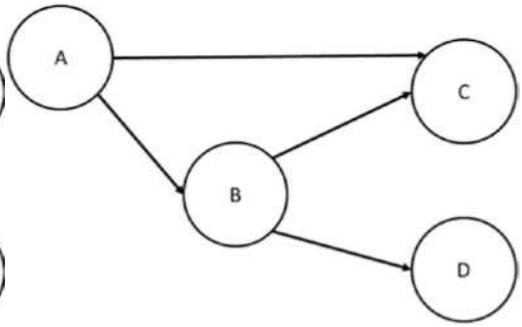

Figure 8-6

A DAG is a data structure that enables you to generate a relationship between data entries that can only be performed in a specific order. The DAG is an important structure in the core data science environments, as it is the fundamental structure that enables tools such as Spark, Pig, and Tez in Hadoop to work. It is also used for recording task scheduling and process interdependencies in processing data.

GML format

The format is simple but effective.

This is a post code node:

```
node [ id 327
label "Munich-80331-DE" routertype
        "PostCode" group0 "DE" group1
        "Munich" group2 "80331"
      ]
```

This is a GPS node:

```
node [ id 328
label "48.1345 N-11.571 E routertype "GPS"
groupO "DE" groupl "Munich" group2 "80331"
      sLatitude "48.1345 N" sLongitude "11.571
      E" nLatitude 48.134500000000003
      nLongitude 11.571
     ]
```

This is an edge connecting the two nodes:

```
edge [ source 327 target 328
    ]
```

Understanding Your Online Visitor Data

Online visitors have to be mapped to their closest billboard, to ensure we understand where and what they can access. To achieve this, I will guide you through a graph data processing example, to link the different entities involved in a graph of the visitor activity.

The data that we have, however, is stored with some issues.

- *The billboard names are missing.* With some feature engineering, we can infer the values from the longitude and latitude values.
- *The distance between billboard and visitor is unknown.* With some feature engineering, via the Vincenty's formulae, this can be found.
- The longitude and latitude requires smoothing, to comply with the billboard naming formatting agreement.

What are Vincenty's formulae? Thaddeus Vincenty's formulae are two related iterative methods used in geodesy to calculate the distance between two points on the surface of a spheroid. They assume that the true shape of Earth is an oblate spheroid and, therefore, are more accurate than methods that assume a spherical Earth. The distance is called the great-circle distance. (See Figure 8-11)

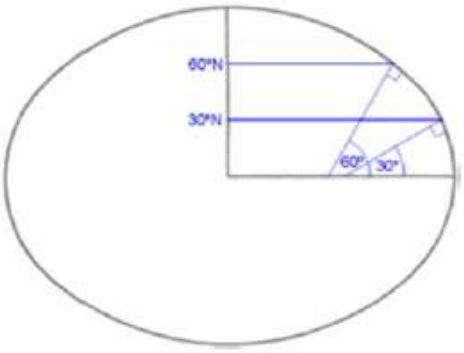

Figure 8-11. *Thaddeus Vincenty's formulae in graphic form*