Dennis Sun **DATA 401:** Data Science Alexander Dekhtyar

Information Integration

Data Integration. Data Integration is the process of combining data residing in different sources and providing the user with a uniform view of this data.

Data Science context. In the context of Data Science, *data integration* can take the following forms:

- **Join.** Different data sources contain *complimentary information* (different features) about the same entities.
- **Union.** Different data sources contain similar information (same features) about potentially different sets of entities.
- **Mix.** Different data sources contain information about potentially different sets of entities, and the information they contain may be different in its nature.
- **Combination.** Different data sources contain information about *different types* of entities (and their relationships).

Examples. Below are examples of each type of the data integration scenarios.

1. Join. A University's Registrar's database contains information about students and the courses they take. The Office of Financial Aid's database contains information about students, their tuition payments and any financial aid students get. Combining these two databases together may allow one to study the relationship between the financial aid provided to students and student performance in coursework.
2. Union. Two grocery store chains merge as a result of corporate acquisition. Both chains have product databases storing information aobut the products sold in the stores. Some products are shared (e.g., products produced by national brands). Some products (e.g., the private labels of each chain) are only found in one database. In order to properly operate the newly merged chain of stores, the product databases need to be merged together.
3. Mix. A health care provider is examining databases of a medical lab and a pathology office. The two databases contain information about the various

tests performed by the medical lab. The pathology database contains extra information about the patient cases (the pathology reports produced by the doctors). At the same time, the pathology office provided its services to more than one medical lab, so some of the cases found in the pathology office database are missing from the medical lab database. The goal of the information integration process here is to merge all matching information from the two databases, while also identifying entities that do not have counterparts in the second dataset.

4. Combination. In order to study voting patterns in the US Presidential election, it may be useful to combine in a single database the following information, currently available from different sources:

   (a) Demographic information about each congressional district and each electoral prestinct. Such information is available from the US Census Bureau.

   (b) Polling data. This information may be available from one of the web sites aggragating various polls.

   (c) Texts of stump speeches of each candidate.

   (d) Relevant tweets.

   Each of these datasets comes from a different data source (US Census Bureau, poll aggregating web sites, candidate web sites, Twitter) and presents data about different entities (congressional districts, individual candidates, individual voters, as representatives of the voting population.).

Each type of integration requires its own approach.

Why Data Integration is Hard?

Data Variety. *Data Variety* affects data integration in a number of ways.

1. Different data sources may represent their data using a variety of different modeling approaches:
   - Relational database (relational tables)
   - Semistructured data (JSON, XML)
   - Unstructured data (text documents)
2. Descriptions of data (database schemas) may be different even when the data is of the same type. (E.g., two relational databases storing corporate employee data for two different companies may have different structure, i.e., different tables, and different attributes in each table).

Data Veracity. Information in the data sources can be:

1. Unreliable. Two different databases may contain different names for the employee with the same SSN.
2. Incomplete. Values of certain attributes may not be available for all entities (or it may simply be impossible to find these attributes).
3. Stale. Certain information may be old and no longer reflective of the current state of the world (e.g., the shipping address of a customer is no longer valid, but it is impossible to determine it).

Algorithmic Complexity. Some of the algorithmic problems arising from data integration needs (e.g., database schema matching, query containment, etc.) have high algorithmic complexity.

Systems. There are two essential ways to approach data integration:

- Distributed query processing/learning. An information need expressed over the overarching data schema is distributed over multiple data sources. The final response is assembled based on the results each data source (and the query/analytical engine assoicated with it) returns.
- Centralized storage of data. Data is transferred from diverse remote sources into the local repository, creating an overarching data model in the process.

Logistical Issues. Logistical issues related with data integration may be broken down into the following aspects:

- Data ownership. Determines whether it is possible to obtain access to the necessary data.
- Data access. The specific means of communicating with the data source (e.g., propriatery API, SQL, etc...).

Completion. Data integration, in many situation, is a *continuous process*, where each subsequent version of the data integration software deployed improves the accuracy and the overall quality of the constructed databases.

Architectures for Data Integration Systems

While individual projects may require unique Data Integration systems built, most Data Integration systems share the overall structure, and share the different component types deployed.

Data Sources. Step one in developing a data integration system is to secure access to each data source. Typical types of data sources are:

- Relational Databases. These are accessed using SQL queries passed to the DBMS where they reside.
- Semistructured data collections. Typically reperesented as XML or JSON object collections. May be deployed inside non-relational DBMS (e.g., MongoDB, MarkLogic, etc), in which case access can be through the query languages employed by those systems (e.g., XPath or JSONPath, or MongoDB's API).
- Streaming Semistructured data. These are usually supported by either proprietary APIs (e.g., Twitter API, Facebook API, Google Maps API), or by web API protocols (SOAP, REST).
- HTML documents. Collections of HTML documents are usually accessed via the HTTP protocol using web crawling and HTML scraping.
- Legacy structured and usntructured data.

Wrappers. A wrapper in the context of a Data Integration system is a component responsible for accessing, extracting, and processing the data from a single data source.

Each data source must have a special purpose wrapper designed for it. Wrappers are tailored towards the specific type of the data source (see above) and the specific needs of the data integration system: i.e., they retreive only the necessary data.

Mediated Schema. A mediated schema is a component of a data integration system that contains the description of the *complete* data collection constructed from the multiple data sources.

The mediated schema of a system usually has the following properties:

- Comprehensiveness. The mediated schema describes **all** the data that is needed for the application.
- Compactness. The mediated schema contains the description of only the data types and features that are needed for the application. Not all data available from the data sources is reflected in the schema.
- Mappings. The mediated schema is supported by the *individual mappings* transforming the data available from individual data sources to the subset of the mediated schema.

Instance constructors. Instance constructors are various components of a Data Integration system whose role is to place the correct data into the current instance of the mediated schema.

To some degree, in a narrow sense of the term *data integration*, the final database instrance constructors are the parts of the system responsible for the data integration per se. These components peroform the following tasks:

- Entity resolution. Determining when two or more instances of data possibly obtained from different data sources contain complementary information about *the same entity.*
- Deduplication. Determining when two or more instances of data (possibly obtained from different data sources) contain duplicate information.
- Merging. Performing the merging of multiple instances of data determined to represent a single entity.
- Feature resolution/Entity representation. Determining the best value for each attribute of an entity given competing values from multiple data sources.
- Concordance construction. In some situations, it is necessary to keep information about all possible values of an attribute in the database instance. This is done by keeping concordance tables, which are build using concordance constructors - components that determine matching values coming from different sources and put together the concordance relationships.

Entity and Attribute Resolution

Entity Resolution problem. Given two or more records obtained from different sources, determine *if these records are descriptions of **the same** entity*, and, if yes, reconstruct, to the best ability, the most complete and the most correct description of the entity.

Attribute Resolution problem. Given two (or more) values of a certain attribute, determine if these values should be treated as equivalent, and if yes, select the best way to represent the given value.

Entity resolution relies in large part on solving the Attribute Resolution problem.

Attribute Resolution

For *numeric* and *date-time* attribute values, attribute resolution boils down to representing the values in the same format and comparing them.

Problems may be encountered due to

- Lack of specificity in the values. (e.g., are "10 million" and 10, 012, 543 to be considered the same value when describing someone's net worth? when describing the available number of observations?)
- Incompleteness of compound values. (e.g., are "10/12/2016" and "October 12" to be considered as the same date or not?)
- Measurement error.

The core problem though, lies in **matching string values**.

String Matching Problem. Given two sets of strings $X = \{x_1, \ldots, x_n\}$ and $Y = \{y_1, \ldots, y_m\}$, find all pairs of strings (x, y), where $x \in X$ and $y \in Y$, such that x and y refer to the same entity (are a representation of the same value).

Example. Consider the following strings: `"Dr. R.S. Murdock"`, `"Robert Murdock"`, `"Dr. L. Moore"`, `"Lauren Moore"`, `"Lori Moore"`, `"Bob S. Murdok"`.

We have reasons to believe that these strings refer to two individuals, Dr. Robert S. Murdock and Dr. Lauren Moore (who sometimes goes by "Lori"). However, in each pair of strings referring to the same individual, each string has some information the other string does not.

Solving String Matching Problem. Typically, to find matching strings, one compares a group of strings to each other using a variety of *string similarity measures.* Based on the observed similarities, *and occasionally based on additional information* (e.g., knowledge of the type of strings that are being resolved), one can make string matching decisions.

The additional knowledge may include:

- Knowing the type of string values being matched: e.g., personal names, addresses, company names, adjective descriptors, and so on.
- Knowing that certain parts of strings are irrelevent for matching purposes (e.g., all strings being compared start with the phrase `"Mailing address:"`)
- Knowing the language(s) used in the strings.

String Similarity Measures.

Multiple String Similarity measures can be used.

Levenshtein (Edit) Distance. The most straightforward measure of string similarity is the edit distance (also known as Levenstein distance), defined as follows:

> Consider a collection of string transformation operation consisting of three operations:
>
> 1. deletion of a letter
> 2. insertion of a letter

3. replacement of a letter with another letter

The edit distance $d(x, y)$ between two strings x and y is the *minimal number* of string transformation operations necessary to convert x into y.

The *edit distance-based similarity* between x and y (or *Levenshtein similarity* is defined as

$$s(x, y) = 1 - \frac{d(x, y)}{\max(|x|, |y|)},$$

where $|s|$ denotes the length of string s.

Edit distance between two strings $x = x_1 \dots x_n$ and $y = y_1 \dots y_m$ can be computed using dynamic programming techniques, via the following recursive relation:

$$d(\epsilon, \epsilon) = 0$$

$$d(x_1 \dots x_i, y_1 \dots y_j) = \min \left(\begin{array}{ll} d(x_1 \dots x_{i-1}, y_1, \dots, y_{j-1}) & \text{if } x_i = y_j \\ d(x_1 \dots x_{i-1}, y_1, \dots, y_{j-1}) + 1 & \text{if } x_i \neq y_j \\ d(x_1 \dots x_i, y_1, \dots, y_{j-1}) & \\ d(x_1 \dots x_{i-1}, y_1, \dots, y_j) & \end{array} \right)$$

Needelman-Wunch score. The Needelman-Wunch score (also known as the global string alignment metric) extends the notion of edit distance by employing a matrix $c(a, b)$ of replacement scores: $c(a, b)$ is the penalty for replacing a letter a with a letter b.

The Needlman-Wunch score of x and y is the smallest score of a global *alignment* between x and y, where the match between each pair of letters is scored using the correspondence table $c(a, b)$, and a match between a letter and a gap is assigned a constant penalty δ (gap penalty).

It can be computed using dynamic programming, via the following recurrence:

$$d(\epsilon, \epsilon) = 0$$

$$d(x_1 \dots x_i, y_1 \dots y_j) = \min \left(\begin{array}{l} d(x_1 \dots x_{i-1}, y_1, \dots, y_{j-1}) + c(x_i, y_j) \\ d(x_1 \dots x_i, y_1, \dots, y_{j-1}) + \delta \\ d(x_1 \dots x_{i-1}, y_1, \dots, y_j) + \delta \end{array} \right)$$

Smith-Waterman Measure. The Smith-Waterman score (the local alignment score) is a variation of the Needlman-Wunch score, that looks for the longest substring match. As such, it "suppresses" the penalties the Needleman-Wunch score would assign for mismatching portions of x and y outside of the longest local alignment.

The Smith-Waterman Measure can be computed using the following recurrence:

$$d(\epsilon, \epsilon) = 0$$

$$d(x_1 \dots x_i, y_1 \dots y_j) = \min \left(\begin{array}{l} d(x_1 \dots x_{i-1}, y_1, \dots, y_{j-1}) + c(x_i, y_j) \\ d(x_1 \dots x_i, y_1, \dots, y_{j-1}) + \delta \\ d(x_1 \dots x_{i-1}, y_1, \dots, y_j) + \delta \\ 0 \end{array} \right)$$

Jaro score. The Jaro measure compares short strings. It works reasonably well on strings representing first and last names, for example.

The Jaro score of two strings x and y is computed as follows:

- Find the sequence of common characters x_i, y_j such that $x_i = y_j$ and $|i - j| \leq \frac{\min(|x|,|y|)}{2}$ (i.e., common characters that are somewhat near each other's positions). Let c be the number of common characters.
- Compare the ith common character of x to the ith common character of y. Compute the number t of positions in which the characters are not equal (i.e. the number of transpositions).
- The Jaro score of x and y, $jaro(x, y)$ is computed as follows:

$$jaro(x, y) = \frac{1}{3}\left(\frac{c}{|x|} + \frac{c}{|y|} + \frac{c - 0.5 \cdot t}{c}\right)$$

Jaro-Winkler score. The Jaro-Winkler score captures cases when the Jaro score of x and y is low, but the two strings have a common prefix.

Let PL be the length of the longest common prefix between x and y.

Let PW be a numeric weight given to the common prefix of x and y.

The Jaro-Winkler score of x and y is:

$$jw(x, y) = (1 - PL \cdot PW) \cdot jaro(x, y) + PL \cdot PW$$

Jaccard and Dice Coefficients. Let x and y be two strings. Let X and Y be *sets* of features extracted from X and Y.

The Jaccard similarity coefficient of x and y is

$$Jaccard(x, y) = \frac{|X \cap Y|}{|X \cup Y|}$$

The Dice similarity coefficient of x and y is

$$Dice(x, y) = \frac{2 \cdot |X \cap Y|}{|X| + |Y|}$$

Features of strings. Jaccard and Dice coefficients ignore the order of characters in a string. But, they allow one to collect multiple different types of features and use those. Possible features are:

- Individual characters.
- Bigrams: sequential pairs of characters in a string. (e.g., the bigrams of "Smith" are ".S", "Sm", "mi", "it", "th", and "h!" where "." stands for "start of string" and "!" stands for "end of string").
- Trigrams or n-grams: generalization of bigrams.
- Words.
- Word stems.

TF-IDF similarity score. The tf-idf similarity score works well with long strings that consist of multiple words.

Let $V = \{t_1, \dots, t_n\}$ be a collection of words that occur in all strings from our set $X = \{x_1, \dots, x_m\}$.

For each term $t \in V$, let $df(t)$ (document frequency of t) be the number of strings $x \in X$ that contain t. Then, let

$$idf(t) = \log \frac{n}{df(t)}$$

For each string $x \in X$ and each term $t_i \in V$, let $tf_x(i)$ be the number of times t_i appears in x (term frequency).

The term-frequency-inverse document frequency vector representation of a string x, $\bar{x}$ is a vector

$$\bar{x} = (w_1, \dots, w_n),$$

where $w_i = tf_x(i) \cdot idf(t_i)$.

The cosine similarity score of two tf-idf vectors $\bar{x}$ and $\bar{y}$ is

$$sim(\bar{x}, \bar{y}) = \frac{\bar{x} \cdot \bar{y}}{|x| \cdot |y|}$$

Entity Resolution

TBA