The data world is having a come-to-Jesus moment. Brave ambitions to derive beautiful insights from previously unmined data sources are forcing us to face the unwieldy results of haphazardly-collected data. Data cleansing absorbs a significant portion of analysis efforts and entity resolution is a major element of data preparation. Reconciliation issues are summarily characterized as below:

* Data is fractured across databases, requiring complex joins,
* Platforms built on different architectures,
* Varying data collection techniques invite errors and
* Original purpose of the data leaves missing, changed, multiple interpretation of values.

Frequently performed manually, entity resolution has quietly evolved into a formal discipline, complete with easily used interfaces[[1]](#footnote-1) and foundational algorithms. This post will focus on the latter of these, summarizing the most popular entity resolution algorithms and their practical implementations.

|  |  |  |
| --- | --- | --- |
| Table 1 ER vs. Clustering | | |
| Algorithm | Entity Resolution | Clustering |
| k-Means | # of clusters is linear  Avg cluster size is constant  Many clusters are singletons | # of clusters is linear |
| LDA[[2]](#footnote-2) |  |  |
| Pair-wise | Match |  |
| Distance |  |  |

Clarifications[[3]](#footnote-3):

ER is not Classification. When we are simply identifying whether or not two records match or do not match we are classifying them. However, entity resolution develops a dynamic entity using metadata. See Table 2 for a comparison.

ER is not Clustering. Differences between the two are itemized in Table 1.

# Underlying Concepts

## Triangular Equality

In this post we will speak in terms of matching between two strings, however, the concept of triangle equality, which addresses equality among three entities is an important consideration. Say we determine that A=B. Separately we determine that B=C. Does it naturally follow that A=C? The truth is, not necessarily. Some algorithms preserve Triangular Equality while others defy it.

Dissimilarity Measure

For every yin there’s a yang. For every similarity, there’s a dissimilarity. When researching, be aware that you may stumble on the yang when you were looking for the yin.

# Let’s Get Started

|  |  |  |
| --- | --- | --- |
| *Table 2 ER Objectives* | | |
|  |  | Examples |
| String Matching | Quantify permutations needed to convert one string to another | Edit Distance, Alignment, Phonetic |
| Distance Metrics | Apply physical distance measures to abstract concept of data objects | Similarity, Text Analytics |
| Relational Matching | Conjunctive view reliant on one data object’s relationship to other objects | Set Based, Aggregate |

While this post is focused on the technical side of algorithms, note that before setting out on your ER task you should first determine which of the main ER objectives is your goal (see *Table 2* for a summary).

Below, for each of the major ER objectives we will:

1. identify specific tasks inside the umbrella objective,
2. define popular algorithms for each task alongside challenges, and
3. show practical packages implementing the algorithm in two programming languages (R, Python)

|  |  |  |
| --- | --- | --- |
| T*able 3 ER vs. Classification [[4]](#footnote-4)* | | |
|  | Entity Resolution | Classification |
| End Entity | Define multi-faceted entities using varying datasets, metadata | Entities are reconciled to each other in an existing static context |
| Single Truth | Relationships are among groups of entities | Pairwise relationships |
| Inputs | Capable of assessing entity across n-number of inputs | Typically matches between two entities at a time |
| Input Sequence | Sequence-neutral; ER models refine entities with new data inputs | Results are sensitive to sequence of when inputs were provided |

## The Data.

We begin with large datasets and apply blocking techniques[[5]](#footnote-5) to reduce the size of probable matches. For two given records within these datasets we begin with the compilation of a set of comparison vectors of similarity scores for component attributes. The score may take many shapes, outlined in *Table 4*.

## The Algorithms: String Matching.

|  |  |
| --- | --- |
| *Table 4 Nature of Similarity Scores in String Matching* | |
| Boolean | 0 or 1; Match or non-match |
| Edit Distance | Quantified permutations to convert one textual string into another |
| Jaccard Coefficient | Ratio of existence or absence of one entity’s individual attributes in another |
| Phonetic Similarity | Pronunciation of letters are phonetically related on a 0 to 1 similarity scale |

Strings are compared using exact element-by-element comparison. Outlined in *Table 4*, there are four essential approaches.

The first, **Boolean**, is easily understood as a Yes or No, 0 or 1, match or non-match between two strings. It is the most simplistic of the group.

The second, **Edit Distance**, is highly studied. Similarity between two strings is quantified by physically measuring the permutations needed to convert one string into other. The core implementation of edit distance is Levenstein, which penalizes for insertions, deletions and substitutions. Over time Levenstein has been modified with additional costs for transpositions, gaps and weighted costs per each of the actions or the location of where the permutation must be made.

The third, **Jaccard Coefficient** (aka, Tanimoto Coefficient as both mathematicians independently founded this ratio unbeknownst of each other), is an element-by-element measure of intersection. Said otherwise, it is the ratio of the intersecting set to the union set. The Jaccard Coefficient satisfied triangle equality.

The fourth, **Phonetic Similarity**, is a throwback to that 80’s-era infomercial, “1-800-ABC-DEFG Hooked on Phonics worked for me!” Phonetic algorithms result in Soundex encodings, which sidestep misspellings and variations, by indexing a table of language-specific homophones for a string’s soundex encoding rather than searching the string itself. Two critical inputs to phonetic similarity are (1) discerning which language the string is written in and (2) knowing the context of the letters you are matching. The crucial former prerequisite is accomplished by matching pronunciation rules of letter sequences using their location in the string (“sch” in German vs. “sz” in Polish at beginning of a string). The latter is accomplished by parsing the string into a sequence of phonetic tokens according to pronunciation rules in that language. The International Phonetic Alphabet (IPA) is popularly used to identify tokens with corresponding sounds, though frequently criticized for being too fine of match.

## The Algorithms: Distance Metrics.

|  |  |
| --- | --- |
| *Table 4 Nature of Similarity Scores in String Matching* | |
| Euclidean | Data objects charted in preference space of graph of x,y axes (common attributes) |
| Manhattan |  |
| Minkowski |  |
| Text Analytics | Jaccard Similarity Coefficient |
| Vector Similarity | Cosine Similarity, TFIDF |

From here we must choose whether to use a Learning- or Non-Learning-Based Matching, detailed in *Table 5*. The former is composed of three essential algorithms: the seminal Fellagi Sunter[[6]](#footnote-6), Cosine, Jaccard and, modified for different applications. The latter approach, as characteristic of machine learning, minimizes human interaction.

|  |  |  |
| --- | --- | --- |
| *Table 5 Learning vs.Non-Learning Metrics* | | |
| Approach | Non-Learning-Based Matching (Probabilistic) | Learning Based Matching |
| General | Edit Distance | 2 phases: model generation, model application |
|  | Single or Conjunctive Match |  |
| Requirements | Similarity Threshold as parameter. Apply threshold-based selection of the matching entity pairs | Model generation needs training dataset w/ manually labeled boolean correspondences |
| Examples | PPJoin+ Cosine  PPJoin+ Jaccard  Fellagi Sunter Trigram  Fellagi Sunter TokenSet  Fellagi Sunter Winkler | FEBRL SVM (Freely Extensible Biomedical Record Linkage)  MARLIN (Multiple Adaptive Record Linkage w/ Induction) has 2 string similarity measures (edit distance, cosine) & several learners (SVM, decision trees) |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  | Overview |  | Challenges |
| 1 | Weighted Sum/ Average & Threshold | Apply threshold to a weighted sum or average of component similarity score | Simplistic | Pick weights, tune thresholds |
| 2 | Rules-Based Matching | Follow defined rules, essentially IIF statements | Simplistic | Manually define rule sets |
| 3 | Fellagi Sunter | How do references co-occur? Generates references from entities | Probabilistic,  Supervised | Supervised |
| 4 | Fellagi Sunter Winkler | Builds on (1) with a latent match variable estimated using Expectation Maximization | Probabilistic,  Supervised | Supervised |
| 5 | Conditional Random Fields | Noun coreference, clique templates w/ tied parameters where decision for one pair affects another through their overlap | Undirected Probabilistic,  Supervised | Supervised |
| 6 | Decision Trees |  |  |  |
| 7 | SVNS |  |  |  |
|  | Jaccard Similarity Coefficient (Index) | Similarity measure = ratio of intersecting set to the union set | Distance metric |  |
|  | Tanimoto | Similarity ratio is same as Jaccard’s, but distance function differs. Allows 2 non-alike specimens to share commonality with a 3rd specimen[[7]](#footnote-7) | Not a Distance metric | Not a proper distance metric as it disproves triangle equality |
| 8 | Ensembles of Classifiers |  |  |  |
| 9 | Weighted K-Partite Matching | Best applied to Record Linkage. Edges are pairs between records from different data sets whose weights are the pairwise match score. | Record Linkage | NP-hard; best to perform successive bipartite matching |
| 10 | Hierarchical Clustering |  | Deduplication |  |
| 11 | k-Nearest Neighbor |  | Deduplication |  |
| 12 | Correlation Clustering | Uses Integer Linear Programming (ILP) to maximize a cost function, placing positive/ negative benefits of clustering mentions of x,y together, accomplishing Transitive Closure | Deduplication | ILP Is NP-hard, requiring heuristics to approximate cost w/ Greedy BEST/FIRST/VOTE, Greedy PIVOT & local search |
| 13 | Edit Distance | Define a most representative centroid @ required permutations among strings | Canonicalization |  |
| 14 | Stanford Entity Resolution Framework (blackbox) |  | Canonicalization |  |
| 15 | LDA |  | Generative Probabilistic, Unsupervised |  |
| 16 | Bayesian Networks |  | Generative Probabilistic, Unsupervised |  |
| 17 | Markov Logic Networks (MLN) |  | Undirected Probabilistic |  |
| 18 | Probabilistic Soft Logic | Reverse predicate equivalence (a friend of a friend might be my friend), predicting match with truth values [0,1]; relaxed declarative language defines continuous constrained Markov random field in 1st order logic, using relaxed logic operators | Undirected Probabilistic |  |
|  | Dice’s Coefficient | Quotient of similarity: # of intersected species / sum of sum total species |  |  |

|  |  |  |
| --- | --- | --- |
|  |  |  |
| Distance Metric | Overview |  |
| Levenstein | Min # of single character insertions, deletions, substitutions to permute one string to another. Possible change to weight cost of insertions/ deletions vs. transpositions |  |
| Damerau-Levenstein | Amends Levenstein to include transposition |  |
| Affine | Amends Levenstein with open gap, extend gap |  |
| Smith-Waterman | Amends Affine with mismatches at beginning/ end lower cost; employs dynamic programming algorithm. More computationally expensive than Levenstein |  |

Appendix

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Concept | Algorithm | Python | R |
| Blocking |  | Hash based  Similarity/ neighborhood  Complex blocking predicates by combining simple  minHash (locality sensitive hashing)  Canopy Clustering |  |  |
| Deduplication | Correlation Clustering | Linear progrmng: max cost fn  Greedy BEST/ FIRST/ VOTE  Greedy PIVOT algorithm  Local search | FuzzyWuzzy  Dedupe |  |
| Canonicalization | Rule Based  Set value attribues | Edit distance  Stanford Entity Resolution Framework (blackbox) |  |  |
| Record Linkage |  | Weighted K-Partite Matching | PyBloom |  |
| Referencing |  |  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| String Matching | Distance Metrics | Relational Matching | Other Matching |
| Edit Distance  - Levenstein  - Smith-Waterman  - Affine | - Euclidian  - Manhattan  - Minkowski | Set Based   * Dice * Tanimoto (Jaccard) * Common Neighbors * Adar Weighted | * Numeric distance * Boolean equality * Fuzzy matching * Domain specific |
| Alignment  Jaro-Winkler  Soft-TFIDF  Monge-Elkan | Text Analytics  - Jaccard  - TFIDF  - Cosine Similarity | Aggregates   * Average values * Min/ Max values * Medians * Mode | Gazettes   * Lexical matching * Named Entities (NER) |
| Phonetic   * Soundex * Translation |  |  |  |

1. D-Dupe is a Python implementation of active learning. [↑](#footnote-ref-1)
2. Latent Dirilect Algorithm [↑](#footnote-ref-2)
3. http://linqs.cs.umd.edu/projects//Tutorials/ER-AAAI12/ER\_Tutorial\_part2.pdf [↑](#footnote-ref-3)
4. http://jeffjonas.typepad.com/jeff\_jonas/2007/09/entity-resoluti.html [↑](#footnote-ref-4)
5. Blocking is not the subject of this paper [↑](#footnote-ref-5)
6. Insert reference to Fellagi Sunter paper [↑](#footnote-ref-6)
7. https://en.wikipedia.org/wiki/Jaccard\_index#Tanimoto\_similarity\_and\_distance [↑](#footnote-ref-7)