**What is data matching?**

The process of comparing two or more records and computing the possibility of them belonging to the same entity.

Firstly predicting the type of columns it has strings, numbers

It also suggestable to know if it is first name or last name or location name or company name

Email idm phone number

Then do the preprocessing which includes cleaning and standardization

* Remove and replace empty values, leading/trailing spaces, specific characters and numbers, punctuation, etc.
* Parse aggregated or longer columns into smaller sub-components, such as Address field into Street Number, Street Name, City, etc.
* Transform letter cases (upper to lower or lower to upper) to ensure consistent, standardized view.
* Merge same or similar columns together to avoid duplicated columns.
* Match and transform patterns of data values for consistency.

**Selecting type of data match algorithm**

Different data matching algorithms are used depending on the nature of the data to be compared. For example, integers are compared differently than open-text string fields, so the entity matching algorithm used to compare names should be different from the one used to compare street numbers. A list of most common data match algorithms used is given below:

**i) Exact:**Does not allow any tolerance in difference and expects to find a full match of character to character.

**ii) Fuzzy:** Allows tolerance in difference and evaluates the likelihood of two strings being similar by using probabilistic matching. [Fuzzy matching](https://dataladder.com/fuzzy-matching-software/) technology utilizes a number of industry leading techniques, such as Levenshtein Distance (or Edit Distance), Damerau-Levenshtein Distance, Jaro-Winkler Distance, Metaphone 3, Name Variant, etc.

**iii) Phonetic:**Useful for matching similar sounding strings as it matches words by their pronunciation. Phonetic match can be exact or fuzzy.

**iv) Numeric:**Used to run a probabilistic match on numeric fields.

**b) Assigning weights to matching attributes**

Some attributes have a higher probability of being unique for an entity as compared to others. For example, it is likely for two customers to have the same name, but unlikely to have the same address as well. Comparing two records with different names but the same address should not compute a 50% match score.

This is where weights can be useful. Weight assignments help to prioritize attributes that are less likely to be the same for two entities and should have higher priority while calculating the match score. In the example we just discussed, assigning a higher weight to the address field makes more sense than the name field.

**c) Thresholding classification rule**

A threshold value defines the classification level for the comparator. If the results of the comparison are equal to higher than the defined level, then the match is considered successful. Any value below the level would be a non-match. For example, if the threshold level is 70, then all match scores above or equal to 70% are a match and the ones below 70 are a non-match.

List the records which are previously rejected earlier then they recorrected it and send for the comparison, find such records which are

Find if there is an exact difference

**1. Exact Matching Algorithms**

These methods check whether two numbers or alphanumeric strings are exactly the same.

* **Direct String Comparison (==)** – Used for exact matching of numeric and alphanumeric values.
* **Hash-Based Matching (e.g., MD5, SHA-256, Bloom Filters)** – Hash values are compared for integrity checks.
* **Database Indexing & Lookup (B-Trees, Hash Indexes)** – Used for exact lookup in large datasets.

**2. Approximate / Fuzzy Matching Algorithms**

These techniques find similarities between numbers and alphanumeric values based on heuristics.

**2.1. Edit Distance-Based Matching**

These methods calculate the number of changes (insertions, deletions, substitutions) needed to transform one number or string into another.

* **Levenshtein Distance** – Measures the number of edits required to make two sequences identical.
* **Damerau-Levenshtein Distance** – Similar to Levenshtein but also accounts for transpositions.
* **Hamming Distance** – Counts the number of differing characters in fixed-length strings.

**2.2. Phonetic & Alphanumeric Matching**

Methods that match numeric and alphanumeric data based on pronunciation or structural similarity.

* **Soundex** – Encodes words into phonetic representations for similarity checking.
* **Metaphone & Double Metaphone** – Advanced phonetic matching algorithms that handle variations in spelling.
* **Cologne Phonetic Algorithm** – Optimized for Germanic languages, but applicable to alphanumeric datasets.

**2.3. Token-Based Similarity**

Breaks numbers or alphanumeric values into smaller components (tokens) and compares them.

* **Jaccard Similarity** – Measures set similarity based on shared characters.
* **Sorensen-Dice Coefficient** – Similar to Jaccard but assigns more weight to shared elements.

**3. Numeric Similarity Algorithms**

Techniques specifically designed for numerical data.

**3.1. Distance-Based Matching**

Measures how close two numbers are.

* **Absolute Difference (|A - B| ≤ Threshold)** – Compares numbers based on a predefined margin.
* **Relative Percentage Difference (|A - B| / max(A, B) ≤ Threshold)** – Useful when matching prices, age, etc.
* **Euclidean Distance (√(Σ(A\_i - B\_i)²))** – Used for multi-dimensional numeric comparisons.

**3.2. Statistical Similarity Measures**

Compares numbers based on probability distributions.

* **Cosine Similarity (cos(θ) = A ⋅ B / (||A|| ||B||))** – Measures similarity in multi-dimensional space.
* **Pearson Correlation Coefficient (r)** – Checks how strongly two numerical variables are related.
* **Mahalanobis Distance** – Used when data distributions are non-uniform.

A screenshot of a computer

Description automatically generated

|  |  |  |  |
| --- | --- | --- | --- |
| **Algorithm** | **Type** | **Works With (Numeric / Alphanumeric / Both)** | **How It Works** |
| **Exact Matching (==)** | Rule-Based | Both | Checks if values are identical character-by-character. |
| **Hash-Based Matching (MD5, SHA)** | Cryptographic | Both | Converts data into a fixed-length hash; matches by comparing hashes. |
| **Levenshtein Distance** | Edit Distance | Alphanumeric | Counts the minimum edits (insertions, deletions, substitutions) required to match two strings. |
| **Damerau-Levenshtein Distance** | Edit Distance | Alphanumeric | Similar to Levenshtein but also accounts for transpositions (swapping adjacent characters). |
| **Hamming Distance** | Edit Distance | Both (Fixed-Length Data) | Measures character-by-character differences but requires equal-length inputs. |
| **Jaccard Similarity** | Set-Based | Alphanumeric | Measures the ratio of shared vs. total unique characters in two sets. |
| **Sorensen-Dice Coefficient** | Set-Based | Alphanumeric | Similar to Jaccard but assigns more weight to shared elements. |
| **Soundex** | Phonetic | Alphanumeric | Encodes words based on phonetic pronunciation to match similar-sounding words. |
| **Metaphone & Double Metaphone** | Phonetic | Alphanumeric | Advanced phonetic algorithm for more precise name matching. |
| **Cologne Phonetic Algorithm** | Phonetic | Alphanumeric | Optimized for European names, used in record linkage. |
| \*\*Absolute Difference (` | A - B | `)\*\* | Numeric |
| **Relative Percentage Difference** | Numeric | Numeric | Matches numbers by computing percentage deviation. |
| **Euclidean Distance** | Numeric | Numeric | Measures the straight-line distance between numeric values in multi-dimensional space. |
| **Cosine Similarity** | Vector-Based | Both | Measures similarity by comparing vector representations of numeric/alphanumeric data. |
| **Pearson Correlation** | Statistical | Numeric | Determines how strongly two numerical variables are related. |
| **Mahalanobis Distance** | Statistical | Numeric | Measures similarity while considering variance in data distribution. |
| **Dynamic Time Warping (DTW)** | Sequence-Based | Numeric | Matches numeric sequences even if they are time-shifted. |
| **Longest Common Subsequence (LCS)** | Sequence-Based | Both | Finds the longest matching subsequence between two data sequences. |
| **Fourier Transform Matching** | Frequency-Based | Numeric | Converts numeric sequences to the frequency domain for pattern recognition. |
| **MinHash & LSH (Locality-Sensitive Hashing)** | Machine Learning | Both | Efficiently finds approximate matches in large datasets using hashing. |
| **k-Nearest Neighbors (k-NN)** | Machine Learning | Both | Finds the closest match based on predefined similarity measures. |
| **Neural Network Embeddings (Word2Vec, FastText, BERT)** | Machine Learning | Alphanumeric | Converts words/names into vector representations to find similar ones. |
| **Autoencoders & Siamese Networks** | Machine Learning | Both | Used for similarity learning by training neural networks to recognize similar patterns. |
| **Rule-Based + ML Hybrid Matching** | Hybrid | Both | Combines domain-specific rules with machine learning to improve accuracy. |
| **Weighted Similarity Scoring** | Hybrid | Both | Assigns different weights to multiple similarity measures for more precise matching. |