The Matching Engine

The Science of Maximising Legitimate Matches, Minimising False Matches and Taking Control of the Matching Process

helpIT systems

CLEANER DATA. BETTER DECISIONS.
The Challenge of Contact Data Matching

Typical success metrics for matching usually revolve around the ‘match rate’ - in other words - how many matches (or duplicates) did you find? Lots of matches are good right? 

*Maybe, but don’t be fooled.* Look a little deeper and you’ll realise that the true measure of success is identifying **TRUE matches** while **minimising FALSE matches** in a sea of inconsistent, incomplete data.

With various data sources, formats, entry points and collection methods, by the time a contact record is added to the database, it’s often corrupted in numerous ways. This makes matching a far greater challenge. Therefore the true challenge of matching is best illustrated by asking this simple question:

**Are these records the same person?**

<table>
<thead>
<tr>
<th>Record</th>
<th>Addressee</th>
<th>Company</th>
<th>Address 1</th>
<th>Address 2</th>
<th>Address 3</th>
<th>Address 4</th>
<th>Postcode</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>A R Shore</td>
<td>Windsor Air Charters</td>
<td>Building 3</td>
<td>25 Kings Mill Lane</td>
<td></td>
<td>Redhill</td>
<td>RH1 5YP</td>
</tr>
<tr>
<td>2</td>
<td>Tony Shaw</td>
<td>WAC</td>
<td>25 Kings Mill Lane, Unit 3</td>
<td>Redhill</td>
<td></td>
<td>Surrey</td>
<td>RH1 5YO</td>
</tr>
<tr>
<td>3</td>
<td>Mr. A Shoer</td>
<td>Windsor Air</td>
<td>255 Kings Mill Lane</td>
<td>Bldg3</td>
<td></td>
<td>Redhill</td>
<td>RH1 5YP</td>
</tr>
<tr>
<td>4</td>
<td>Anthony Shaw</td>
<td>Windsor Air Charters, Ltd.</td>
<td>Bldg 3, 25 Kings Mill Ln</td>
<td></td>
<td></td>
<td>Redhill</td>
<td>Surrey</td>
</tr>
</tbody>
</table>

You would probably say that these are indeed the same person. But you might be surprised to learn that with most matching engines — *this duplication would go completely undetected*. Factor in the average rate of duplication in a database of even just a hundred thousand records and the problem is significant.

Let’s look at why...
Most matching software identifies duplicate records by using a specific combination of match keys such as:

**SURNAME** (FIRST FEW LETTERS OF) and/or **FORENAME** (INITIAL OF) and/or **PREMISE** NUMBER and/or **STREET** (FIRST FEW LETTERS OF) and/or **POST CODE** and/or **TOWN**

Just based on these types of match keys, the records on the previous page might never be considered as potential matches.

<table>
<thead>
<tr>
<th>Postcode + Surname (first 5) + Premise</th>
<th>Postcode + Surname (first 5) + Forename (first 5)</th>
<th>Surname (first 5) + First Initial + Premise + Street Name (first 5)</th>
</tr>
</thead>
<tbody>
<tr>
<td>RH15YPSTONE25</td>
<td>RH15YPSTONEA</td>
<td>SHOREA25 KINGS</td>
</tr>
<tr>
<td>RH15YOSHAO 25</td>
<td>RH15YOSHAO TONY</td>
<td>SHAW T25 KINGS</td>
</tr>
<tr>
<td>RH15YOSHOTER255</td>
<td>RH15YOSHOTERA</td>
<td>SHOERA255KINGS</td>
</tr>
<tr>
<td>SHAW 25</td>
<td>SHAWANTHO</td>
<td>SHAW A25 KINGS</td>
</tr>
</tbody>
</table>

If your contact level match key is, say, Postcode (first 5) + Surname (first 5) + Forename (first 5) you will miss matches where one record is for John Smith and the other for J Smith or where the postcode differs. On the other hand, if your match key just includes the initial of first name, you will wrongly match John Smith and James Smith at person level, but still miss matches where one record has a forename of Bill and the other just an initial W for William. Of course, if the name field can’t be parsed into title, Forename and Surnames, or the street address parsed into premise number and street name, then even these crude match keys won’t be available to you.

While common algorithms (such as Soundex, Jaro-Winkler, Levenshtein, N-Gram and Metaphone) do provide some ability to locate miskeyed variations, these solutions all miss potential matches: for example, where there are phonetic differences, missing data, or where the address can’t be standardised and contains non-standard or misplaced elements. Typically, they also give either black and white answers (match or no match) or very crude percentage match numbers – this means that they either return a high number of false matches or miss a large number of matches, depending on the match keys or threshold you choose.

**WHY COMMON MATCH STRATEGIES FAIL**

1. They don’t attempt to understand the pronunciation of the name,
2. They do not allow for inconsistent, non-standard or transposed data,
3. They do not allow for missing or incomplete data, and
4. Matches are not graded, so if you want to avoid false matches, you have to review all of the matches found.
In order to effectively match contact data, you need an engine built from the ground up to deal with the variations found in matching name and address inputs. You also need something to effectively match other contact attributes such as email address, phone number, company and date of birth, which pose different challenges. The matchIT API is specifically designed to deal with these challenges. By **comparing all relevant data** and using a **variety of strategies** to deal with the wide array of hearing, reading, keying and “lack of standards” errors typically found in contact records in most databases, matchIT is able to achieve **results that are almost human in perception**.

### Why matchIT succeeds...

<table>
<thead>
<tr>
<th>Strategy</th>
<th>Benefit</th>
<th>Example</th>
</tr>
</thead>
</table>
| **PARSING & RESTRUCTURING**     | identifies and relocates data to its correct columns, splits unstructured names into component parts | • relocate names in address lines and/or postcodes in the town field, etc.  
• splits town, county, and postcodes combined in one field  
• splits prefix and forename or surname/last name |
| **PHONETIC MATCHING**           | converts names, companies and street/place names into sophisticated phonetic equivalents that compensate for the various spellings and misspellings of similar sounding names | • Shore and Shaw (sound the same)  
• Silva and Silver (sound similar)  
• Ball and Bailey (sound different but would be considered a match by some pseudo-phonetic matching algorithms) |
| **ELEMENT MATCHING**            | to match names with elements missing or reversed                        | • Dr Kiran Patel, Kiran Patel DCh and Patel Kiran |
| **NAME LEXICONS**               | allow for the prevalent use of nicknames                                 | • Bill and William, Jose and Pepe, Elizabeth and Betsy |
| **ACRONYM AND INITIAL MATCHING**| match inconsistencies between names arising from use of initials, abbreviations or acronyms | • The Department for Work and Pensions and DWP  
• Tony and A. (e.g. Tony Leighton and Mr. A. Layton) |
| **BUSINESS WORD IDENTIFICATION**| utilises specific logic to detect the primary words and understand the similarities between words, particularly important for business name matching | • Legal & General Assurance Society Limited matches Legal & General  
• Auto and Motors are equivalent  
• Bijouterie and Jewelerie are equivalent |
| **NON-PHONETIC FUZZY MATCHING** | detects and resolves common keying errors and transpositions and errors in reading handwritten forms | • Wilson, Wislon and Wilsn  
• Morton and Horton |
| **STANDARDISED STRINGS AND WORDS** | match data in different languages, strings abbreviated in different ways | • The Hague and den Haag  
• Brussel & Bruxelles  
• Street & St, Straße & Str. |
So how would matchIT find these matches?

The matchIT API encourages use of multiple match keys so that it *doesn’t rely on any single item of data always being consistent and present*. The keys are tight enough to make sure that the volume of comparisons is reasonable, but loose enough to catch the records with typical errors and inconsistencies. In this example, typical match keys will result in records being compared as follows:

### Key 1: Records 1 & 4 and 2 & 4 will be compared

<table>
<thead>
<tr>
<th></th>
<th>First Name</th>
<th>Last Name</th>
<th>Company</th>
<th>Address</th>
<th>City</th>
<th>Postcode</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>AR</td>
<td>Shore</td>
<td>Windsor Air Charters</td>
<td>Building 3</td>
<td>25 Kings Mill Lane, Unit 3</td>
<td>Redhill</td>
</tr>
<tr>
<td>2</td>
<td>Tony</td>
<td>Shaw</td>
<td>WAC</td>
<td>25 Kings Mill Lane, Unit 3</td>
<td>Redhill</td>
<td>Surrey</td>
</tr>
<tr>
<td>3</td>
<td>Mr. A Shoer</td>
<td>Windsr Air</td>
<td>255 Kings Mill Lane</td>
<td>Bldg3</td>
<td>Redhill</td>
<td>RH1 5YP</td>
</tr>
</tbody>
</table>

### Key 2: Records 1 & 2 will be compared

<table>
<thead>
<tr>
<th></th>
<th>First Name</th>
<th>Last Name</th>
<th>Company</th>
<th>Address</th>
<th>City</th>
<th>Postcode</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>AR</td>
<td>Shore</td>
<td>Windsor Air Charters</td>
<td>Building 3</td>
<td>25 Kings Mill Lane</td>
<td>Redhill</td>
</tr>
<tr>
<td>2</td>
<td>Tony</td>
<td>Shaw</td>
<td>WAC</td>
<td>25 Kings Mill Lane, Unit 3</td>
<td>Redhill</td>
<td>Surrey</td>
</tr>
</tbody>
</table>

### Key 3: Records 1 & 3 will be compared

<table>
<thead>
<tr>
<th></th>
<th>First Name</th>
<th>Last Name</th>
<th>Company</th>
<th>Address</th>
<th>City</th>
<th>Postcode</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>AR</td>
<td>Shore</td>
<td>Windsor Air Charters</td>
<td>Building 3</td>
<td>25 Kings Mill Lane, Unit 3</td>
<td>Redhill</td>
</tr>
<tr>
<td>3</td>
<td>Mr. A Shoer</td>
<td>Windsr Air</td>
<td>255 Kings Mill Lane</td>
<td>Bldg3</td>
<td>Redhill</td>
<td>RH1 5YP</td>
</tr>
</tbody>
</table>

### Match Keys Used*

- Surnames are a phonetic match
- Street name is parsed out & results in a match
- Shore and Shaw get the same phonetic name
- First half of postcode is parsed and results in a match
- Postcode is the same

*The data shown in red forms the basis of the match key for that comparison.

**ACCURACY MEETS SPEED**

These match keys run quickly and efficiently on large volumes of data, comparing only records with enough in common to warrant a closer look, not reporting those pairs which don’t reach the specified threshold, and ignoring records which don’t have enough similarity. It’s smart, it’s quick, it’s scalable and above all, it’s effective.
Fuzzy Matching (and Grouping) in Action

When these different pairs are compared, the records in each pair score highly enough to be reported, when using default name and address matching weights.

<table>
<thead>
<tr>
<th>Addressee</th>
<th>Address 1</th>
<th>Address 2</th>
<th>Town</th>
<th>County</th>
</tr>
</thead>
<tbody>
<tr>
<td>A R Shore</td>
<td>Building 3</td>
<td>25 Kings Mill Lane</td>
<td>Redhill</td>
<td></td>
</tr>
<tr>
<td>Tony Shaw</td>
<td>25 Kings Mill Lane, Unit 3</td>
<td>Redhill</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mr. A Shoer</td>
<td>255 Kings Mill Lane</td>
<td>Bldg3</td>
<td>Redhill</td>
<td></td>
</tr>
<tr>
<td>Anthony Shaw</td>
<td>Bldg 3, 25 Kings Mill Ln</td>
<td>Redhill</td>
<td>Surrey</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Postcode</th>
</tr>
</thead>
<tbody>
<tr>
<td>RH1 5YP</td>
</tr>
<tr>
<td>RH1 5YO</td>
</tr>
<tr>
<td>RH1 5YP</td>
</tr>
</tbody>
</table>

Fuzzy Matching Techniques

- Leighton sounds like Layton
- Tony is recognized as a nickname for Anthony
- Shoer is a transposition of Shore
- 25 is a fuzzy match of 255
- The building number is in different columns and abbreviated differently, but it is all identified as 3
- The street is Kings Mill Lane and the town is Redhill - remember, the matchIT API compares address lines as a block, not simply address 1 with address 1, etc.
- The postcode is RH1 5YP or RH1 5YO, a transposition of that, or is missing. Missing data doesn’t necessarily stop two records from matching

GROUPING

All four records can now be amalgamated into the same match set by grouping pairs together, having used just three match keys. Although you can use as many match keys as you want, which could trigger comparison of records 2 and 3, and 3 and 4, this isn’t necessary as record 1 bridges the other three. Additional data attributes such as email and/or phone can also be included in either key or weighting stages to complement the name and address grouping.
Unlike competing applications and simple database queries which are prone to delivering false matches, matchIT uses a proprietary scoring mechanism that allows you to automate matching based on your own parameters controlling what constitutes a match and what doesn’t. Whenever matchIT compares any two records, each item or group of items compared is given a score which then rolls up into a cumulative score for the entire match.

These match scores allow the engine to determine which matches are likely to be true, which matches are likely to be false and which matches are too close to call. Using a scoring methodology is the only way to get true control over matching and automate the decisioning process.

**Score 90+**

**TRUE Match**
*Automate merge or linking process*
Set your score threshold where you are confident records are **TRUE** matches.

**Score 75-89**

**GRAY Area**
*Requires human review?*
You can choose to review these records individually, put them aside (for a subsequent project), ignore them or (in the case of a cold mailing), remove.

**Score 74 & Below**

**NOT a Match**
*Do not merge or link.*
Set your score threshold above matches that are too inconsistent to be considered duplicates and eliminate **FALSE** matches.

---

**The Importance of Scoring**

<table>
<thead>
<tr>
<th>Record</th>
<th>Name</th>
<th>Company</th>
<th>Address 1</th>
<th>Address 2</th>
<th>Town</th>
<th>County</th>
<th>Postcode</th>
<th>Total Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>AR Shore</td>
<td>Windsor Air Charters</td>
<td>Building 3</td>
<td>25 Kings Mill Lane</td>
<td>Redhill</td>
<td>RH1 5YP</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Mr. A Shoer</td>
<td>Windsor Air</td>
<td>255 Kings Mill Lane</td>
<td>Bldg3</td>
<td>Redhill</td>
<td>RH1 5YP</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Match Score</td>
<td>40</td>
<td>0 or 40*</td>
<td>21</td>
<td>30</td>
<td>91</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*0 when matching at individual level, 40 when matching at business level.
Let’s go back to our original four records. These are fuzzy matches. Often, matches are more obvious, but still are not detected by other software.

Fortunately, matchIT will find them, give you a choice and put you in control. Here’s how...

1. **The matchIT API matches entire records** - using all available data to determine potential matches
2. **matchIT does not rely on extended match keys** that are prone to missing matches and delivering false positives
3. **matchIT uses multiple sophisticated approaches** to ensure that differences arising from all these causes are identified - ultimately finding matches that would otherwise go undetected
4. **matchIT intelligently scores matches** to confidently determine which records are a true match and which records are NOT

Ultimately, the true value of a matching engine is measured by how many true matches and how many false matches it finds on your own data – which only a fully-featured evaluation or equivalent trial can determine.

matchIT captures the highest number of accurate database matches of any data-cleansing solution I have tried.

-Graham Maher, Commercial Analyst, Blackburn Rovers FC
The helpIT matching engine is the core to all of helpIT systems’ data quality applications. Whether you have millions of contacts stored in a database or have dozens of contact data sources that you need to integrate, validate and link in a meaningful way, the exceptional match capabilities of helpIT’s core API can be leveraged through one of our data quality tools. This allows its users to adopt consistent and highly effective rules both to clean and standardise existing data, and to ensure that new data entering the database is of the same high standard.

**Intelligent Matching in SQL Server**
High-performance data quality solution tightly integrated within Microsoft SQL Server, allowing you to run sophisticated matching, cleansing, address validation and deduplication processes on large volumes of data with unprecedented match results.

**Sophisticated Point-of-Capture Duplicate Prevention**
Leverage the strength of matchIT Web to prevent duplicates plus capture valid addresses at point-of-entry in a web form, CRM or other ERP system using sophisticated autocomplete and intelligent inquiry technology.

**Unprecedented Desktop Deduplication and Matching**
Easy-to-use desktop data quality software for batch cleansing of records, sophisticated enough for large volume, automated processing but simple enough for a non-technical user.

Most data quality solutions are limited to simply checking the boxes. That is, they make shallow improvements to the data but never improve the quality enough to offer any real business value. helpIT caters to an audience that recognises the value of accurate matching and is looking for a real-world solution to establish and maintain honestly clean data.
Don’t just check the boxes. Get Cleaner Data with the Right Matching Engine.

Time and again in competitive analysis, the matchIT API finds more true matches with fewer false matches than any other solution on the market. If you want genuinely clean data, choose a solution that deals directly with the nuances of contact data matching and will allow you to easily identify those hard-to-find duplicate records.


For the past 20 years, helpIT systems has been tightly focused on developing and delivering data quality technology that generates tangible and accurate results. With over 1,500 clients in 25 countries across 5 continents, helpIT is consistently raising the bar on data quality success.

Learn more and request a FREE TRIAL at www.helpIT.com.

UK, Europe & Asia: 44 (0) 1372 360070
Australia & New Zealand: 02 8036 3191
The Americas: 866.332.7132
sales@helpIT.com / support@helpIT.com