*Title:* Who’s Who?: Using Entity Resolution to Untangle Identities

*Author:* Sara Kelemen

We’ve all been there—trying to look someone up online. Maybe a classmate, a celebrity crush, a mutual friend. You search their name, and suddenly you’re met with a dozen slightly different versions: one with a middle name, one in all caps, one with a typo that makes you question everything. Maybe on Instagram I’m “Sara Kelemen,” in academic publications I’m “Sara Kate Kelemen,” and someone, somewhere, has undoubtedly written “Sarah Keleman” (the correct spelling is without an H—if you have an H, we’re enemies). Despite their inconsistencies, all of these names represent the same person: me. And figuring that out is the job of entity resolution—the process of determining when multiple records, often messy and imperfect, refer to the same real-world identity.

This isn’t just a problem for curious internet sleuths. In large datasets like those used in healthcare, banking, or government systems, entity resolution is essential for connecting information accurately. I recently tackled this challenge using National Provider Identifier (NPI) data, where I attempted to match doctors to organizations in an entirely separate dataset. The catch? There were no shared unique IDs, and the names almost never matched cleanly.

In the NPI data, I had to handle a wide range of inconsistencies. A single doctor might appear under a full legal name in one place, a nickname or set of initials in another, and a typo-ridden version somewhere else. Organizations weren’t much better—they often came with strange formatting, cryptic abbreviations, or inconsistent punctuation. The lack of a common identifier between the two datasets meant that the only possible link between records was the names themselves, and those names couldn’t be trusted to match exactly. My job was to make those connections anyway—to identify when a doctor in the NPI database corresponded to a patentee, even when their names looked nothing alike at first glance.

To tackle the issue, I built a name-matching pipeline using Python and SQLite. At the heart of it was a simulation process. Since I didn’t have a massive labeled dataset of confirmed matches to train on, I created one. I took real names from both the doctor and patentee tables and introduced random errors—swapping letters, deleting characters, or making substitutions—to mimic the kind of typos and inconsistencies I’d seen in the wild. For example, “Sara Kelemen” might become “Sra Keleemn.” While these corrupted versions may have been frustrating to read, they were incredibly helpful in teaching a model what “almost the same” looks like.

Once I had this simulated dataset, I calculated similarity metrics for each name pair. I used two different string distance algorithms: Jaro-Winkler, which favors names that are similar in spelling and order, and Levenshtein distance, which counts how many edits—insertions, deletions, or substitutions—are needed to turn one string into another. These scores, calculated for both first and last names, gave me a numeric sense of how closely two names resembled each other. Then, I trained a Random Forest classifier on those similarity scores. This model learned to distinguish between “matches” and “non-matches,” recognizing patterns in how names tend to differ when they still refer to the same person. It learned which kinds of typos were forgivable and which ones were deal-breakers.

Of course, comparing every single doctor to every single patentee would be computationally expensive and wildly inefficient. To make this more scalable, I implemented a technique called blocking. In my case, this meant only comparing names that shared the same first letter of the last name. So “Kelemen” and “Keller” might be worth checking, but “Kelemen” and “Gomez”? Probably not, intuitively. It’s a rough cut, but one that significantly reduces the number of comparisons without dramatically hurting accuracy.

After calculating the similarity scores and making predictions with the trained classifier, I stored the matched pairs in a new SQLite table. This table functioned as a bridge between the two original datasets, linking doctors and patentees that may have been written differently but, according to the model, likely represented the same entity. The end result was an efficient and scalable system for making sense of inconsistent name data using lightweight tools and interpretable logic.

Entity resolution might sound like a technical chore, but it’s a powerful tool for ensuring that our data actually reflects reality. Whether it’s doctors and organizations, customers and accounts, or just different versions of your own name floating around the internet, it all comes back to the same question: who’s who? And in a world full of typos, nicknames, and abbreviations, the ability to answer that question accurately makes all the difference.