

Endowing Reasoning Capabilities to a Matching/Scrubbing Algorithm

Jean-Patrick Tsang, PhD & MBA (INSEAD) and Igor Rudychev, PhD
Baysier Consulting
August 16, 2002

Journal of Data Warehousing, October 2002

Abstract

Data matching/scrubbing tools are amazing in that they can perform an astronomical number of matches in a very short period of time. If we forget about sheer volume, however, man beats computer hands down when it comes to subtle situations. The human mind is immensely creative and resourceful and will nail down situations the computer program misses unwittingly. This should be a big surprise! Not that we are smart, of course, but that the computer program is not up to snuff, especially in an era where the most powerful chess player is a computer program. In this article, we offer practical how-tips on how to beef up the quality of your data matching/scrubbing.

Introduction

Consider the following matching/scrubbing problem where two databases of people names – with no common id – are to be combined into one. One database may refer to Bob Smith while the other to Smith, Robert, Jr. Since Bob is short for Robert, those two records may be pointing to the same person. This can be ascertained if other pieces of information such as address, city, state, and zip jibe. In another instance, one database may talk about Chris Black while the other Christine Black-Miller. Once again, we may be facing the same-person situation: maiden name in one case and married name in the other. However, we would rule out that possibility in a split of a second if the latter database also contains a Christopher Black. This would lead us to conclude Chris Black is a he, not a she. These two simple examples should convey the spirit of the problem, although real-life matching/scrubbing undertakings are far richer and spin countless variations on a good handful of recurring themes.

Engage people in the process of matching and you will observe two things. First, you'll be pleasantly amazed how resourceful they are at disambiguating equivocal situations. They effortlessly draw upon a host of tricks: judgment, rules of thumb, common sense, cues, cultural conventions, etc. They notice patterns in the data, craft lines of reasoning, and come up with astute answers. Second, they quickly lose interest in the process and get bored.

How do commercial tools stack up against their homo sapiens counterpart? Man beats machine hands down as soon as the situation gets a little tricky, let alone complicated. And this holds true for all the commercial packages we've looked at. No-brainer

situations on the other hand remind us how unsuited we are for the job in contrast with the machine’s wondrous ability to process millions of records and never getting tired.

Is this suggesting a trade-off between the ability to process large volumes of data and the sophistication with which data can be handled? Certainly not! If current matching/scrubbing tools tend to have bird-size brain capability, this is certainly not due to a sluggish state of the art. Indeed, it’s been years since Deep Blue plays better chess than any mortal including the phenomenal Kasparov. Without a doubt, we are entitled to expect much more from matching/scrubbing tools. This article attempts to reduce that man-machine chasm by describing seven simple do-it-yourself ways to boost the reasoning power of your matching/scrubbing systems (see Table 1).

1. Propagation Triangles and Squares

Just because each element of the address is correct does not mean the address is correct, as exemplified by the following: “Las Vegas, NY 89109” and “Skokie, IL 60016”. Option 1 consists of ignoring the problem, which is wrong. Option 2 recognizes there is a problem without attempting to resolve it, which is better. Option 3 attempts to fix the problem and needless to say this is the one we recommend. To that end, we recommend using propagation triangles and squares.

Consider “Las Vegas, NY 89109”. Picture a triangle where the city, state, and zip are the vertices of that triangle. Las Vegas is in agreement with 89109 while NY disagrees with both Las Vegas and 89109. Said differently, Las Vegas and 89109 support each other while NY is by itself. To disambiguate this situation, we call upon a principle we call “Minimum Goof-Up”. Simply put, it states that when errors occur, they tend to be minuscule or present themselves in small numbers, contrary to the infamous Murphy’s law. The “Minimum Goof-Up” principle in this case suggests it is NY that is wrong because it is the only term not to be supported at all. As a result, the correct address is “Las Vegas, NV 89109”. Note the typo has to do with the fact that the letters V and Y look alike to the hasty person.

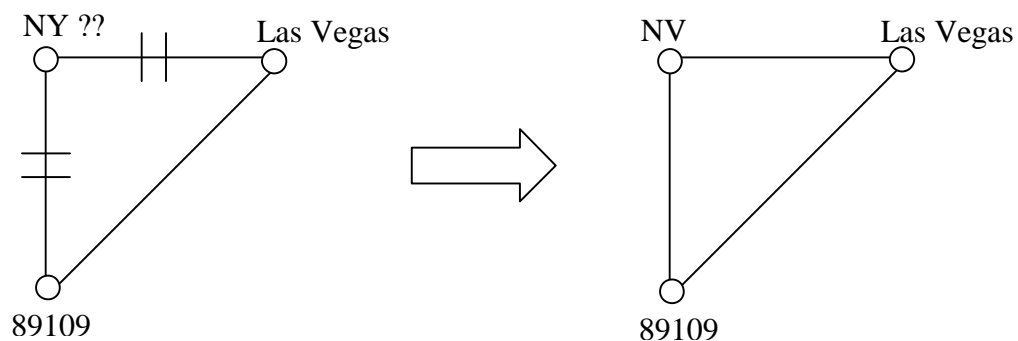


Figure 1: The propagation triangle above suggests NV instead of NY

Let's turn to a seemingly similar example: "Skokie, IL 60016". This time, the propagation triangle suggests the state (IL) is supported both by the city (Skokie) and the zip (60016), and disagreement occurs between the city and the zip. The Minimum Goof-Up principle is inconclusive in this case since the culprit can either be the city or the zip. One may even argue the culprit is more likely to be the zip than the city since it is easier to mistype a zip than substitute Skokie for Des Plaines, the city that contains 60016. To do justice to this case, we bring in the street address, hence the propagation square (the address is the fourth vertex). It turns out that the address "4709 Golf Road" is in agreement with the state and the city, but not with the zip. The verdict is definitive: it is the zip that is wrong and the correct address should read "4709 Golf Road, Skokie, IL 60076". Note the typo lies in the fourth digit of the zip: a "1" was entered instead of a "7".

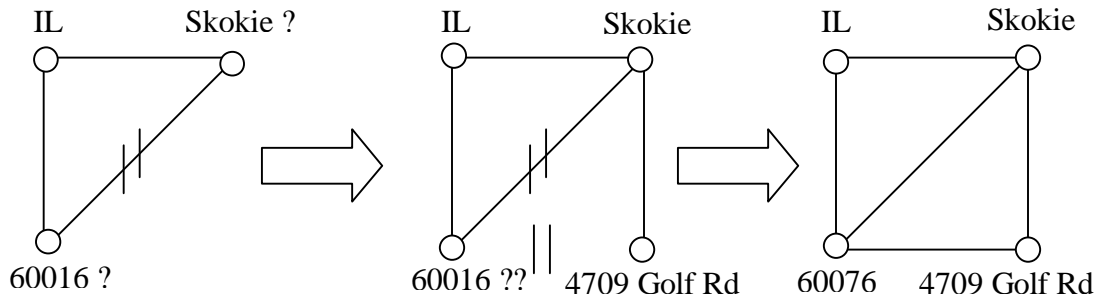


Figure 2: The propagation square above suggests 60076 instead of 60016

Propagation triangles and squares are helpful not only because they help detect contradictions but they also suggest which element of the address is likely to be the culprit.

2. Use Dictionaries

When it comes to matching/scrubbing, we are undoubtedly "unconscious competent". Indeed, we do not even realize we draw upon large bodies of knowledge that we started to acquire since early childhood. We effortlessly recognize Mike is short for Michael, Philly stands for Philadelphia, UCLA represents the University of California, Los Angeles, and st, str, strt are mere abbreviations of street.

If we genuinely want the program to emulate us, one of the first things we need to do is to impart that knowledge to the program. Use dictionaries profusely! We recommend theme-variation dictionaries whereby multiple variations are associated to one theme. For instance, Bob, Bobby, Bobbie, Rob, Bert will all point to Robert. Have the program look up a dictionary of first names, a dictionary of facility names, a dictionary of street types, a dictionary of city names and places, a dictionary of state names, a dictionary of

zip codes, etc. in the matching process, and you will be rewarded with a quantum jump in the performance of the program.

You can even take this one step further. Say you are unclear if the name in question follows a “first name – last name” or a “last name – first name” format. Here is a technique we recommend. Run a frequency analysis of first names and last names by tallying the number of times a first name or last name appears. If you are processing physicians, analyze first names and last names of physicians as opposed to first names and last names of the general US population. Short of such a source, you can use first names and last names of the database you are processing provided you unambiguously know which is which. Now that you have completed the frequency analysis, compute the likelihood of say John Smith where John is the first name and Smith is the last name by multiplying the frequency of occurrence of John as the first name by the frequency of occurrence of Smith as the last name. Similarly, compute the likelihood of the alternate situation (Smith as first name and John as last name). You can safely conclude the one with the higher score is the correct one as long as it is one order of magnitude larger.

3. Use Abstraction Operators

You may be tempted to use theme-variation dictionaries to capture typos. Our experience reveals this can quickly get out of hand since the number of misspellings is potentially very large and as a result any one version of the dictionary may be sinning by omission. In one database alone, we came across 17 different spellings of the city name “Albuquerque”.

We recommend the use of abstraction operators. In a nutshell, they extract the essence and drop the superficial, here the typo. Consider the operator that drops vowels and repetitions of consonants. Applied to Cincinnati (many people get confused as to the number of n’s and t’s), that operator yields “CNCNT”, thereby screening the most common typos.

In the same vein, the Soundex operator helps handle last names. You may recall Soundex was the brainchild of Margaret Odell and Robert Russell at the National Archives in the 1880’s in an attempt to group phonetically similar names for the US Census. The coding starts with the first letter of the last name, and maps the next three consonants onto numbers 1 to 6 based on the following: B, P, F, V = 1 -- C, S, G, J, K, Q, X, Z = 2 -- D, T = 3 -- L = 4 -- M, N = 5 -- R = 6. As an example, the Soundex code for Tsang is T252 and Rudychev R321.

Another useful operator is one that generates initials. The comparison is then performed on the initials. This comes in handy when one database refers to say NYU, VA MC, U Mass, and JF Kennedy while another to NY University, Veterans Affairs Medical Center, University of Massachusetts, and John F. Kennedy, or even New York Univ, Veteran Affairs MC, Univ of Mass, and JFK. The initials generated here are NYU, VAMC, UM, and JFK.

4. Incorporate Plan B

Consider the Double Tree Hotel that stands next to our office building. It is at the crossroad of Skokie Boulevard and Golf Road. Because Skokie Blvd is a larger artery than Golf Road, its address is 9599 Skokie Boulevard. Nonetheless, a piece of mail addressed to Double Tree Hotel at 4707 Golf Road (one building north of 4709) will reach its destination just as well. Question for you: Have you ever been rankled by the fact that a building located at the crossroads of two important streets, like in the Double Tree example above, takes on one street address in one instance and the other street address in another instance?

One way to handle such a situation is to geo-code the two addresses and measure the distance between the two. A short distance may be suggesting you are in the situation above. While this is an elegant situation, it defers the decision making to a later stage in the process, which may translate into reduced efficiency. For this reason, we recommend the concurrent use of an early-stage decision-making technique that turns out to be quite helpful. In a nutshell, that technique consists of assigning to any record not one but two addresses and as a result issuing a favorable verdict whenever either one address of one record matches either one address of the other record. In the very likely event there is only one address, the second address placeholder is simply left blank and the whole processing takes place as if there were only one address to start with.

Parenthetically, we use that same principle to handle first names. This is because a first name abbreviation, say, Bert, may stem from multiple first names (e.g., Robert, Albert). Although one can argue we should remember not two but potentially all related first names, our experience suggests two is the right number. Indeed, two strikes a good balance between complexity and relevance. Besides, the first two options tend to cover more than 80% of the cases.

5. Aggregate Scores Non Linearly

Assume for the sake of simplicity each record is made up of six components only: first name, last name, street address, city, state, and zip. To compare two records, one typically computes six scores (one for first name, one for last name, etc.) and aggregates them into a single score. Aggregated adequately, that single score truly captures the similarity between the two records.

Simplicity suggests a weighted sum. This means assigning, for instance, a 10% weight to the first name score, a 20% to the last name score, a 20% for the address score, a 10% to the city score, a 20% to the state score, and a 20% to the zip score. If the results are unsatisfactory, this simply suggests an adjustment of the weights is in order and the problem will be promptly resolved. Right?

Not quite! Actually, things are a little more complicated. The fact that two first names are identical has no bearing if the last names are different (assume we are not in a situation where a woman is taking her husband's name). Likewise, the fact that two addresses are the same is immaterial if they are from different cities, states, or zip codes. Clearly, the simple weighted sum is inadequate. What we recommend is upgrading this scheme to include conditional weighting. In the case of the first name for instance, define the first names weight to be say X% of the last name score. This will ensure that the first weight will be high if the last names are the same, and low if they are different.

6. Quality Interaction with the User

Rule out prompting the person who runs the fuzzy matching program with questions as the processing unfolds. Instead, keep all the questions for the end. This will save money and make a happier user of the software. Indeed, he/she will no longer have to be tied up to the computer while the program runs, awaiting the next question which may not come up before quite some time in some cases. Savings come from the fact that the freed up user (typically, a layperson with no particular expertise in data quality) can attend to other tasks while the program runs. When it comes to the Q&A session, the user will find the interaction more stimulating and as a result more productive because the downtimes will have been eliminated.

In the Q&A session, make sure the machine does not conduct an interrogatory. The last thing the person attending to the software wants is a pop-up window that reads "Question 527 out of 2,356: Does 4707 Golf Road, Skokie, IL 60076 designate the same entity as 9599 Skokie Blvd, Skokie, IL 60077?". Instead, give that person the upper hand, so questions may be browsed, grouped by issue type, and sampled on the fly. For instance, let the user express the following: "Display cases where one address is correct and the other is syntactically accurate but does not appear in the dictionary of addresses". Reinforce this principle by offering to display charts and graphs pertaining to relevant summary statistics (e.g. frequency distribution of successful and unsuccessful matches by score ranges).

When it comes to asking questions to the user, volunteer peripheral and background information along with the question. Oftentimes, they provide clues to the person attempting to answer the question. Say the issue at hand is whether two addresses represent the same entity. Instead of displaying the two addresses, present a map of the neighborhood with the two locations clearly geo-coded and mention the crow-fly distance between the two locations. The program can even skip the question if the distance exceeds say one mile, saving the user the hassle of saying no.

7. Provide Explanation Capabilities

Encourage the person who uses the fuzzy matching programs to conduct overlap analyses to question the results the tool comes up with. For instance, “Why are those two records matched?” or “Why are those two records left unmatched?”. Developing the critical mind has three benefits.

First, it allows the user to identify shortcomings in the tool. Asking why the record that involves Dick was not matched with the one that involves Richard may lead to the realization that the Dick-Richard pair was omitted in the first names dictionary, a blemish that can promptly be remedied. More subtly, it may suggest it is how the overall fuzzy score is pieced together that leaves to be desired. In this case, the record pairs prompting the question may provide insights regarding the range of values the aggregate fuzzy function should be returning and, as a result, what changes need to be brought about.

Second, having the option to question any verdict pronounced by the tool and reverse that decision dissipates the uneasiness of using a piece of software, especially blackbox programs. This puts the user and not the program in the driver’s seat.

Third, confidence in the tool is reinforced whenever the answer under scrutiny proves to be correct. Indeed, going through the auditing process and realizing one’s hunch was wrong lends even more credibility to the tool. The user is reminded that, like the machine, she follows inference rules but, unlike the machine, she makes mistakes the machine never makes.

Conclusion

Implementing the seven techniques described above will immediately bring about three benefits in addition to a significant improvement in the data quality. First, the cost of scrubbing will be significantly reduced. Indeed, more iffy cases will be resolved by the machine, thereby cutting back on costly human intervention to resolve issues unattended by the software. Second, inaccuracies and gawkiness in the results will point to gaps in the tool that can be addressed right away, for instance, incomplete dictionaries, equivocal arbitration rules, and outright oversights. Needless to say taking care of those issues will in turn squeeze the cost of data scrubbing even further. Third, cases left unresolved by the tool will be smaller in number but larger in difficulty. This heightened focus on the key issues is bound to be more inviting for the analysts to embrace and, as a result, to develop appropriate solutions for.

- Use a slew of Dictionaries (first names, cities, states, zips, addresses, street types, etc.)
- Make sure natural “constraints” among the various elements of the address (e.g. street address, city, state, and zip code) are satisfied.
- Use abstract versions of names (last names, company/organization names) when carrying out comparisons.
- Expand the matching universe by carrying a second alternative (e.g. Bert can be short for Albert but also for Herbert).
- Build a fuzzy score that acknowledges dependencies among the various elements of the address. For instance, that the first name is the same is immaterial if the last name is different (for women, the situation is a little more subtle).
- Put the end-user of the matching/scrubbing tool in the driver’s seat. The program one should never forget is the slave, not the master.
- Discourage sheepish acceptance of the results - Encourage active questioning of the results. This is key in taking the program to the next level.

Table 1: Key ideas to boost performance

Jean-Patrick Tsang is the founder and president of Bayser, a Chicago-based consulting firm dedicated to sales and marketing for pharmaceutical companies. In a previous life, JP studied Artificial Intelligence and developed several techniques to automate tasks such as process planning of mechanical parts, design of payloads for satellites, and architecture of cargo ships and cruise-liners. In addition to a PhD in Artificial Intelligence from Grenoble University, JP has an MBA from INSEAD in Fontainebleau, France. He can be reached at (847) 920-1000 or bayser@bayser.com.

Igor Rudychev is a senior consultant with Bayser. Prior to joining Bayser, Igor has been conducting research in optimization techniques applied to super string theory, the most promising avenue to the unification of quantum mechanics and Einstein’s theory of gravity. Igor has a PhD in Theoretical Physics from Texas A&M. He can be reached at (847) 679-8278 or igor@bayser.com.