Detecting duplicates of individuals in Icelandic legacy data

Jón Steinn Elíasson

Reykjavik University
jone20@ru.is

February 14, 2021

Abstract

Identifying by name brings a lot of problems such as typos and conventions. This paper is about the detection of duplicate individuals in an old Icelandic legacy database. We use a distance measure, domain specific string manipulation as well as comparing other fields to detect duplicates. The number of duplicates found amounted to 16.63% *of the total rows.*

I. Introduction

The best way to avoid duplicates is not to produce them to begin with. That can be easier said than done as we might have limited control over data in the wilderness. Redundant data increases as data grows and given its volume today, the need for data cleaning has never been greater.

The legacy database we look at is from an Icelandic volleyball association where records were manually entered. Existing individuals were looked up by name and birthday to determine if they already existed. If not a new individual was created.

Even in a country as sparsely populated as Iceland and its small subset of people associated with volleyball one cannot assume uniqueness of names. Adding date of birth might suffice (for this scenario, not generally) but one never knows beforehand and there are so many issues associated with such identifiers. Berman [1] goes over a few such as that names can change and special typographic characters and probably the most common one, the human error factor.

We apply similarity matching and domain specific string manipulation to try to detect as many duplicates as possible with the aim of minimizing false positives.

The rest of this paper is structured as follows. Chapter II defines our similarity measure. In chapters III and IV we describe our implementation and experimental setup. Potential improvements are listed in Chapter V and finally we interpret the results in Chapter VI.

II. Background

Almost all Icelandic names end with "son" or "dóttir" meaning son and daughter respectively. Instead of surnames we append those endings (depending on sex) to the first name of a parent (usually the father). Sharing such an ending (especially for females) can skew a similarity function greatly.

There are various ways of measuring similarity between words. One such method is the *Levenshtein distance* [2] which measures how many character insertions, deletions or substitutions are required to change one word into another. The more similar the words the less the Levenshtein distance. If the strings are equal, no operations are needed. If not, we can remove excess characters from the longer word and replace every remaining character to match the other word, thus the length of the longer word is an upper bound on Levenshtein distance. Let ω_1 and ω_2 be two non-empty words of length $|\omega_1|$ and $|\omega_2|$ respectively and let $\mathsf{lev}(\omega_1, \omega_2)$ be their Levenshtein distance. We can then map the similarity into probability with $\mathsf{lev}_p(\omega_2, \omega_1) = 1 - \frac{\mathsf{lev}(\omega_1,\omega_2)}{\max\{|\omega_1|,|\omega_2|\}}$.

III. Methods

We begin by grouping the individual according to their first letter in lower case where some Icelandic letters are mapped to their English counterparts as shown in Table 1. This is done

á	é	í	ó	ý	ð	ö
a	e	i	o	y	d	o

Table 1: *The mapping of some Icelandic letters to their English counterparts.*

to reduce the number of comparisons needed assuming typos are rare in the first letter of a word.

For each group we compare every pair and assess whether they are the same person or not. If a pair is considered to be the same person, we connect them in a graph data structure so we can easily extract them as a group later.

The process of determining each pair starts with mapping the names to lower case and according to the map in Table 1. This is done to catch typos and informal writing styles. We also remove any "son" and "dottir" endings as they will skew the similarity function towards deciding on a match. Let ω_1 and ω_2 be the pair's names after the mapping. We set an initial threshold for the similarity function to a value $T \in [0, 1]$. If the pair shares a

T	Initial threshold
B	Same birthday
E	Shared email
P	Shared phone number
N	Matches nickname
I	Same initials

Table 2: *Summary of variables. Their meaning or condition is on the right side.*

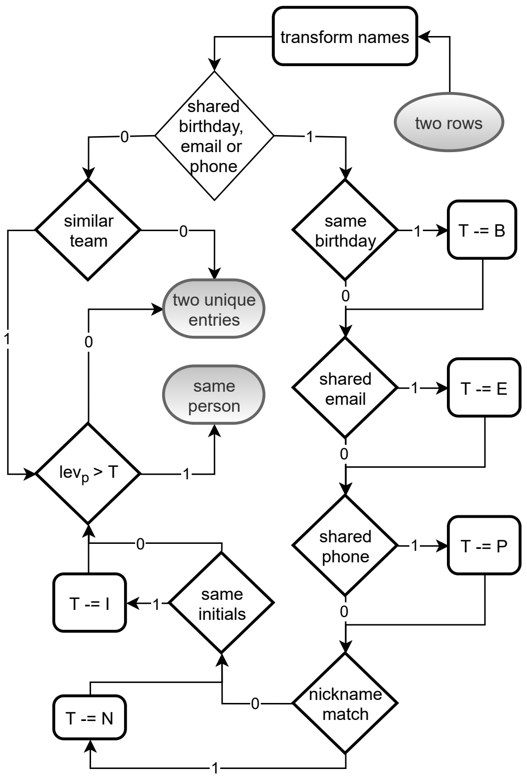

Figure 1: *Flowchart of how the decision of determining if two rows are the same real life person or not. It shows how the threshold (T) can be lowered by various factors. Boolean values are represented with 0 and 1.*

birthday, email or phone number we reduce the threshold by B, E and P respectively. If none of these are shared and their teams are not similar (with a high threshold) we decide the pair is not the same person regardless of the name. If birthday, email or phone number is shared we further check if the first name of one matches a set of predefined nick names for the other and if their initials match, reducing the threshold by N and I respectively. Finally we decide that the pair is actually the same person if $\mathsf{lev}_p(\omega_2, \omega_1) > T$. If not they are considered as two unique individuals. This process is summarized in Figure 1.

After going through all pairs in all groups we have a set of graphs connecting all occurrences of individuals that appear more than once. We do not actually change the database but rather create a map from the original row to its duplicates. The timestamp column is used to decide who is the original.

IV. Results

The script was written in Python using the `pandas` and `textdistance` libraries to create data frames and calculating Levenshtein distance respectively. The assignment of the variables defined in Section III that we used can be seen in Table 3.

T	B	E	P	N	I
0.95	0.1	0.15	0.15	0.25	0.25

Table 3: *The assignment of variables used in the script.*

The sizes of the groups of duplicates can be seen in Figure 2 but these groups include the original.

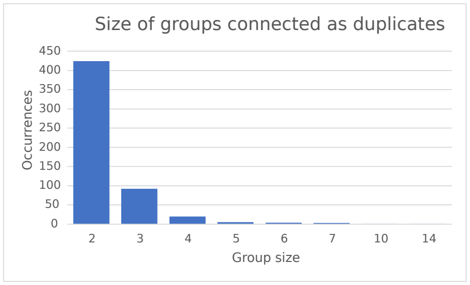

Figure 2: *The distribution of the size of duplication groups, each group being a single real life individual.*

The total number of duplicates found was 748. This number is excluding the original rows from each group of connected duplicates. That amounts to 16.63% of all rows.

V. Future work

There is always room for improvements, both in reducing false positives and better ways of detecting duplicates. There are namely two methods we wanted to try.

We are not utilizing the other tables at all. They contain a lot of information that we can use to avoid false positives. Suppose two rows we think are the same person play for different teams in the same tournament or one is maybe a referee and the other one playing. That makes them an unlikely candidate for being the same actual person.

Hagstofa Íslands is an Icelandic public institution that tracks various statistics, one being how many people have specific names, either as first or second names. For example, there are a lot of people with Jón as their first name but very few with the first name Þormóður. We could scrape this data and make the threshold depend on name rarity.

VI. Conclusions

It is difficult to measure our success without knowing which rows are duplicates. The grouped rows were skimmed over to see if any false positives had occurred but none were spotted. If that is indeed the case, then any reduction of data without removing information can be considered a success.

Considering this database was in actual use it is quite astonishing it has 14 rows of the same individual. It underlies the need for cleaning data and what problems identifying by name can bring.

Regardless of identifiers, there will always be errors when someone is manually writing to databases. We could use a combination of fields with at least one unique field such as email but that does not guard against human errors. We could provide a suggestion of matches for the person typing them in, similar to how we are finding duplicates here, where he can choose to add to an existing individual or create a new one.

References

[1] Jules Berman. *Principles and practice of big data : preparing, sharing, and analyzing complex information*. London: Academic Press, 2018. Chap. 3, pp. 53–69. ISBN: 978-0-12-815609-4.

[2] Vladimir Iosifovich Levenshtein. “Binary codes capable of correcting deletions, insertions and reversals.” In: *Soviet Physics Doklady* 10.8 (Feb. 1966). Doklady Akademii Nauk SSSR, V163 No4 845-848 1965, pp. 707–710.