Detecting duplicates of individuals in Icelandic legacy data

Jón Steinn Elíasson

Reykjavik University
jone20@ru.is

February 7, 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. We assume that existing individuals were looked up by name 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 the 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 goes over a few in [1] such as names can change and special typographic characters and probably the most common one, the human error factor.

We apply some similarity matching and some 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 chapter III we describe our implementation. In chapter IV we specify the experimental setup and finally interpret the results in Chapter V.

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. 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 birth-

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.*

day, 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) in either case. 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 nor decide which row is kept.

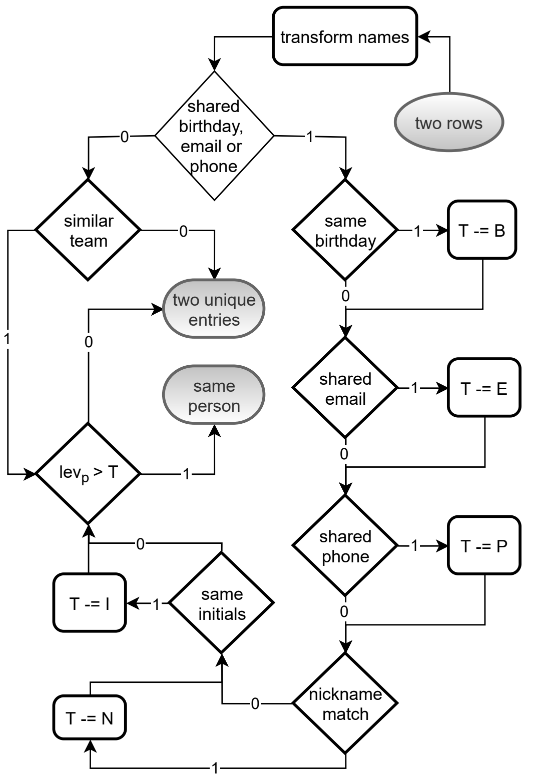

Figure 1: *Flowchart of how the decision of determining if two rows are the same real life person or not. It shows how the threshold (T) can be lowered by various factors. Boolean values are represented with 0 and 1.*

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 will include all individuals having at least one duplicate. If we were transforming the data we would want to keep one and update rows in other tables for the other ones.

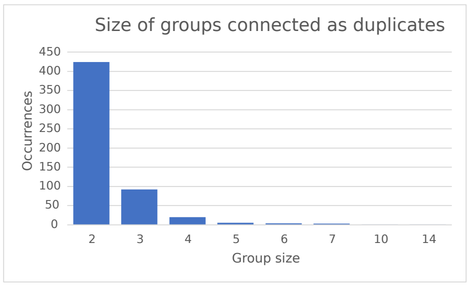

Figure 2: *The distribution of the size of duplication groups, each group being a single real life individual.*

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

V. 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 and date of birth can bring.

References

[1] Jules J. Berman. *Principles of Big Data: Preparing, Sharing, and Analyzing Complex Information*. 1st. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2013. ISBN: 9780124045767.

[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.