Census Index Merge

By Jim Wray of MyFamily.com

Problem Description and Motivation: The process of keying censuses to create a digital index is error prone, difficult, and expensive. Once an investment is made into that process one would like to leverage that investment to its fullest extent. Given two large indexes keyed at different times, of the same information, the desire would be to take advantage of both indexes and enhance the probability of a user finding their ancestor. For the purposes of this paper, we will discuss the merging of the US 1920 census indexes.

Complexity: The every-name version (I_E) of the 1920 census index is approximately 107 million names and the head-of-household version (I_H) is about 36 million names. The simplest algorithm is of the order $O(n^2)$ which for this case would be

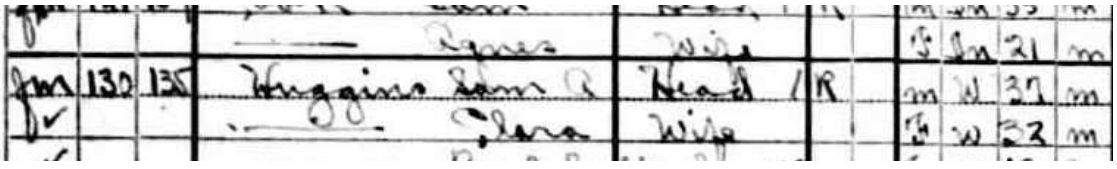
$$107{,}000{,}000 \times 36{,}000{,}000 \approx 3.9 \text{ quadrillion}$$

record comparisons. To put that in perspective, say a machine could do ten thousand compares a second. It would then take more than 12,000 years (on one machine) to complete the calculations. A priori knowledge of the data lets us restrict our space on both sides at the very minimum to approximately $n/2076$ because there are 2,076 microfilm rolls which we consider hard barriers that the data doesn't cross. So, if we use 107,000,000/2,076 and 36,000,000/2,076 as our new numbers in this simple algorithm, we can expect approximately

$$52{,}000 \times 17{,}000 \times 2{,}076 \approx 1.8 \text{ trillion}$$

compares as a worst case scenario. Further, significant restrictions to the search space can be affected by first testing common header and page information such as county, locality, enumeration district, and page number. Because of the tendency for the header information to be consistent and more correctly keyed, the method allows us to reduce complexity significantly; however, this effect cannot be quantified well given that there is no guarantee that the information (other than the roll) is the same between the two indexes.

Example: Below we present an example of the 1920 US Census.



The (pertinent) keyed information for this image from the two indexes:

Index	Surname	Given	Gender	Race	Age
Head of Household	Wiggins	Sam A	Male	White	37
Every Name	Huggins	Sam A	Male	Indian[1]	37

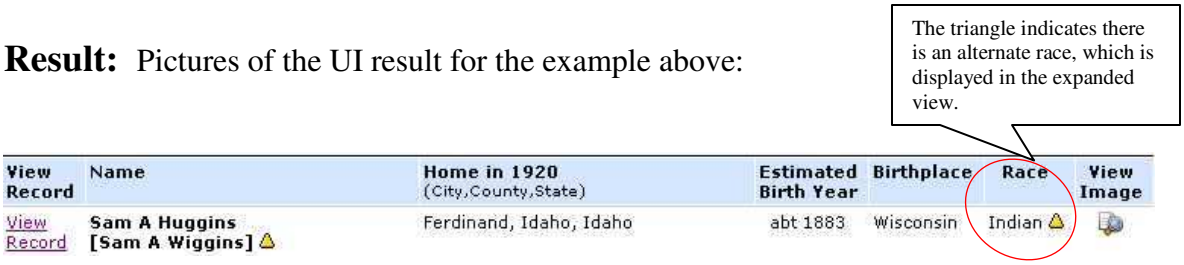
Table 1: Results from the two different keying instances.

This example illustrates an instance where two different surnames were keyed. It is difficult know for sure which keyed surname is correct. Suppose the correct keying is "Huggins," then the "H" appears unusual with the curvature near the top and the low cross-bar. However if "Wiggins" is the correct keying, then "i" is unusual because of the lack of a dot and the way the enumerator came out of the "W" which appears to be a "u." Further, commonality of surname is of little help as Wiggins is only twice as likely as Huggins.[2] If instead, we explore the possibility of keying the surname as literally as we can decipher, a keyer might choose to key this as "Wuggins." But since this is an unknown surname, the likelihood of that being the correct interpretation is miniscule.

Method: We employ the Levenshtein distance[3] or "edit distance" to give a closeness measure between textual fields in the indexes. We also use special mixes of Levenshtein and integer differences to determine scores for numerical fields (such as age). Finally, we normalize and combine the scores in a weighted fashion where the result lies between zero and one. We then iterate through the current set (narrowed down as much as possible) and record the best score for each name.

Once the best matches are determined and recorded along with their match score, a new index needs to be created that includes alternate names that can be searched and displayed within one record. For high confidence matches, we group and index both names, and if there is any indicator of uncertainty in one or the other index, we use the more certain name as the default. For any perfect matches (exactly 1.0) we do nothing[4]. For low confidence matches, we discard the match and defer to the newer index. In all cases, where uncertainty is equal between indexes, we defer to the newer index.

Result: Pictures of the UI result for the example above:



[1] The "Indian" keying comes from a carry-over mistake – the individual above is Indian.
[2] From final counts of the keying results in the 1920 Census of Ancestry.com
[3] V. I. Levenshtein. *Binary codes capable of correcting deletions, insertions and reversals*. Doklady Akademii Nauk SSSR **163**(4) p845-848, 1965,
also Soviet Physics Doklady **10**(8) p707-710, Feb 1966.
[4] We are considering making this a high priority match for global search because it is a confirmed, double-keyed name.