# RecordLinker

RecordLinker is a novel record linking method. The basic steps to generating Unique Historical Individuals are described below. The process is described in detail afterwards.

1. Preprocessing data in pairs.py  
   The input data provided by ELO is processed into pairs.
2. Generating links in recordlinker.py  
   Similar pairs are found by comparing names. The person references in these similar pairs are linked.
3. Processing links to Unique Historical Individuals in "Unique Individuals/individuals.py"   
   The linked person references are grouped to find all person references that refer to the same person.

## Pairs

In pairs.py, the following input files are read and processed:

1. Geboorte.csv
2. Huwelijk.csv
3. Overlijden.csv

These files contain birth, marriage and death certificates. Each certificate mentions people. These people can be divided into pairs. The pairs describe a man and a woman. The following types of pairs are possible:

|  |  |  |  |
| --- | --- | --- | --- |
|  | Person 1 | Person 2 | Certificate type |
| **1** | father | mother | birth |
| **2** | groom | bride | marriage |
| **3** | father of groom | mother of groom | marriage |
| **4** | father of bride | mother of bride | marriage |
| **5** | deceased | partner | death |
| **6** | father of deceased | mother of deceased | death |

(The sex of the deceased and their partner is not known. This is important for the order of the name comparison)

Also, for some types of pairs, the name of a child is provided. Children need to be incorporated into the RecordLinker to find the birth certificate of Unique Historical Individuals. The pairs with children become as follows:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Person 1 | Person 2 | Child | Certificate type |
| **1** | father | mother | newborn | birth |
| **2** | groom | bride | - | marriage |
| **3** | father of groom | mother of groom | groom | marriage |
| **4** | father of bride | mother of bride | bride | marriage |
| **5** | deceased | partner | - | death |
| **6** | father of deceased | mother of deceased | deceased | death |

The result of pairs.py is a .csv file with all pairs. It contains the following fields:

|  |  |
| --- | --- |
|  |  |
| **year** | Year the event took place: yyyy |
| **first\_letters** | First letter of last name man + first letter of last name woman |
| **pair** | Type of pair (integer)  **1:** parents of newborn  **2:** groom and bride  **3:** parents of groom  **4:** parents of the bride  **5:** deceased and partner  **6:** parents of deceased |
| **man** | Name of the man |
| **woman** | Name of the woman |
| **child** | Name of the child  "" if no child is provided |
| **age** | Either the age of the bride or groom, depending on the type of pair:  age of bride if pair type is 1 or 3  age of groom if pair type is 2  -1 if age is unknown or pair type is not 1, 2 or 3 |
| **uuid** | uuid of the certificate |
| **man\_uuid** | uuid of the person reference of the man |
| **woman\_uuid** | uuid of the person reference of the woman |
| **child\_uuid** | uuid of the person reference of the child  "" if no child is provided |

Note that for pair type 5 (deceased and partner) the sex is unknown. The deceased will be treated as the man and their partner as the woman. This causes an issue in the linker. The order of the names affects the Levenshtein distance. The RecordLinker has a function to minimize this impact.

## Name Processing

The input data contains fields for first name, prefix, last name, age, place of birth, date of birth, occupation, place of residence and comments for each person reference. The names are processed in the following way:

1. Only the first name and last name fields are used (no prefixes)
2. Each first name and last name are cleaned up individually
3. The cleaning up removes capital letters, removes accents and other diacritics, only keeps symbols in the alphabet and replaces
   1. ch for g
   2. c for k
   3. z for s
   4. ph for f
   5. ij for y
4. Orders first names alphabetically and appends the last name (no spaces)

**Example:**

Mariä Anña van 't Schip → annamariasgip

Also, the first letter of the last name is saved (used to optimize algorithm)

## RecordLinker

Afbeelding met Lettertype, tekst, Graphics, schermopname

Automatisch gegenereerde beschrijving

The RecordLinker finds similar pairs and saves these links. The RecordLinker has 14 modes. These modes describe what type of pairs are compared. A mode describes the type of reference pair and the type of pair that are potential links. For example, mode 1 has reference type 1 and potential links 1. Type 1 of pairs are parents of newborns. This means that mode 1 links parents of newborns with each other.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Person 1 | Person 2 | Child | Certificate type |
| **1** | father | mother | newborn | birth |
| **2** | groom | bride | - | marriage |
| **3** | father of groom | mother of groom | groom | marriage |
| **4** | father of bride | mother of bride | bride | marriage |
| **5** | deceased | partner | - | death |
| **6** | father of deceased | mother of deceased | deceased | death |

The created pairs above are compared in recordlinker.py. Each type of pairs is compared to all other pairs, including the same type. An exception to this are married couples. Married couples are not compared to married couples, because the same couple can only marry once.

Each type of comparison is configured as a linking mode. A mode describes the type of reference pair and the type of pair that are potential links. These modes are displayed below:

|  |  |  |
| --- | --- | --- |
| Mode | Reference pairs | Potential link pairs |
| **1** | Parents of newborns | Parents of newborns |
| **2** | Parents of newborns | Parents of deceased |
| **3** | Married couples | Parents of newborns |
| **4** | Married couples | Parents of married couples |
| **5** | Married couples | Parents of deceased |
| **6** | Parents of married couples | Parents of married couples |
| **7** | Parents of married couples | Parents of newborns |
| **8** | Parents of married couples | Parents of deceased |
| **9** | Deceased and partners | Parents of newborns |
| **10** | Deceased and partners | Married couples |
| **11** | Deceased and partners | Parents of married couples |
| **12** | Deceased and partners | Deceased and partners |
| **13** | Deceased and partners | Parents of deceased |
| **14** | Parents of deceased | Parents of deceased |

As seen, only 5 different type of pairs are listed although in pair.py 6 types were declared. This is because for the most part, pair 3 and 4 are treated the same. These go together as parents of married couples. The schema for pairs becomes:

|  |  |
| --- | --- |
| Type of pair | Name |
| **1** | Parents of newborns |
| **2** | Married couples |
| **3,4** | Parents of married couples |
| **5** | Deceased and partners |
| **6** | Parents of deceased |

For a clearer overview of the pairs the same mode table is displayed below with the pair numbers:

|  |  |  |
| --- | --- | --- |
| Mode | Reference pairs | Potential link pairs |
| **1** | 1 | 1 |
| **2** | 1 | 6 |
| **3** | 2 | 1 |
| **4** | 2 | 3,4 |
| **5** | 2 | 6 |
| **6** | 3,4 | 1 |
| **7** | 3,4 | 3,4 |
| **8** | 3,4 | 6 |
| **9** | 5 | 1 |
| **10** | 5 | 2 |
| **11** | 5 | 3,4 |
| **12** | 5 | 5 |
| **13** | 5 | 6 |
| **14** | 6 | 6 |

Example:

RecordLinker in mode 3 links ‘Married couples’ (type 2 pairs) to ‘Parents of newborns’ (type 1 pairs)

Linking

The recordlinker compares reference pairs to potential link pairs. Reference pairs are all pairs of the desired type. Potential link pairs are pairs of the desired type that have the same first letters as the reference pair. Also, potential link pairs are sometimes filtered on year depending on the mode.

Each potential linking pair is iterated over and the Levenhtein distances of the names are evaluated. If the distance is smaller or equal to the maximum accepted distance, the two pairs are saved as a link.

Child linking

-

Year filtering

-

Cpu\_boost

-

Special rules

Cant link the same pair

Type 5 unknown sex