RECORD LINKAGE PROBABILÍSTICO APLICADO A CENSOS DEMOGRÁFICOS

Valentin Vinagre Urteaga

Resumen— Las fuentes históricas de naturaleza demográfica permiten estudiar no solo el comportamiento demográfico, si no también entender la evolución social y económica del pasado. Para que toda esta información sea útil en investigaciones, es necesario realizar una vinculación de estos datos, pero a la hora de realizar vinculaciones de censos y padrones históricos existen varias dificultades. Esta documentación histórica viene de diferentes fuentes y épocas, por tanto, ni los nombres, apellidos, direcciones, etc están siguiendo un mismo estándar y realizar una vinculación directa no es posible. Los algoritmos de recordlinkage son los intentan vincular toda esta información teniendo en cuenta toda la variabilidad y la dificultad con la armonización de los datos.

En este artículo se ha realizado una propuesta de mejora en el algoritmo Record Linkage de tipo probabilístico actualmente utilizado en el CED (centro de estudios demográficos), sin tener en cuenta la vinculación de los familiares.

Palaras clave—Python, record Linkage, probabilístico, red social, censos, padrones, demográficos, CED, vinculaciones.

Abstract— The historical sources of demographic nature allow to study not only the demographic behavior, but also to understand the social and economic evolution of the past. In order for all this information to be useful in research, it is necessary to link these data, but when making linkages to censuses and historical registers, there are several difficulties. This historical documentation comes from different sources and times, therefore, neither the names, surnames, addresses, etc. are following the same standard and making a direct link is not possible. The recordlinkage algorithms are the ones trying to link all this information taking into account all the variability and difficulty with the harmonization of the data.

In this article we have made an improvement proposal in the Record Linkage algorithm of probabilistic type currently used in the CED (center of demographic studies), without taking into account the connection of the relatives.

Index Terms— Python, record Linkage, determinist, social network, censuses, demographics, CED, linkages

1 INTRODUCCIÓN

Actualmente existen muchos documentos históricos digitalizados, que necesitan ser vinculados para que investigadores de diferentes campos puedan utilizarlos en sus estudios e investigaciones.

Este trabajo se ha desarrollado para complementar el proyecto de investigación XARXES1, el cual tiene como objetivo crear redes sociales históricas basadas en la vinculación de censos y padrones con la colaboración del CED (Centro de estudios demográficos) [1] y del CVC.

El segúndo paso para construir una red social es poder vincular la información contenida en los registros de los diferentes censos. Dado que toda esta información no está estandarizada no se puede vincular directamente y se necesitan de algoritmos más complejos que den una solución a la variabilidad de los datos, para permitir realizar una vinculación. Además, existen más dificultades añadidas, por ejemplo: cambios de apellidos, nacimientos, defunciones, etc.

Estos problemas se vuelven más desafiantes en los primeros censos históricos, es decir, los recogidos en el siglo XIX o principios del XX, donde solo se disponía de información limitada sobre individuos. Como resultado, vincular individuos no es confiable, y a menudo se generan muchas coincidencias falsas o duplicadas.
El CED está utilizando un algoritmo que vincula mediante los datos de nombres y apellidos diferentes registros de censos y padrones, además de realizar vinculaciones de familiares. Lo que se presenta en este trabajo es la realización de un algoritmo record linkage que permita la vinculación de censos mediante más campos.

2 OBJETIVOS

El principal objetivo de este proyecto es la realización de un algoritmo probabilístico capaz de enlazar censos patrimoniales mejorando el utilizado actualmente en el CED. Para ello se han realizado los diferentes objetivos:

- Enlazar personas que aparecen en los censos.
- Realizar un estudio sobre las distintas soluciones

- *E-mail de contacto: valentin.vinagre@e-campus.uab.cat*
- *Mención realizada: Ingeniería de Computación*
- *Trabajo tutorizado por: Alicia Fornés Bisquerra*
- *Curs 2018/19*

sobre record linkage.
- Proposición de un método dado los pros y contras de este y los estudiados.
- Testeo de la solución para comparar si los resultados son mejores que lo implementado actualmente.
- Estudiar una manera de interactuar con el algoritmo por parte del usuario.

Como subobjetivos se han realizado los siguientes puntos a lo largo de todo el estudio:

- Realizar un adecuado diseño del programa para así otorgarle legibilidad, funcionalidad y flexibilidad.
- Estudiar e implementar una arquitectura de datos que permita que el algoritmo desarrollado funcione lo más rápido posible.
- Investigación sobre distintas librerías de código compartido, se debe tener en cuenta que la comunidad somos todos y siempre está bien contribuir para la mejora de código y funcionalidades.

3 ESTADO DEL ARTE

Se han realizado varias investigaciones sobre estudios orientados al tratamiento de distintos problemas RecordLinkage y a continuación se resumen los más significativos:

- Automated census record linking: a machine learning approach [2].

Solución pensada solamente para personas masculinas, debido al cambio de apellidos de las personas femeninas.

Pros: Varias opciones a la hora de realizar enlaces de nombres utilizando distintos tipos de distancias (jarowinkler, soundex, etc) para cálculos de distancias y numeraciones, explicaciones de construcción de variables útiles a la hora de realizar el algoritmo, el algoritmo realizado para el sexo masculino podría ser útil en ciertos matices.
Contras: El algoritmo utilizado no plantea el sexo femenino, por lo que no es del todo válido en ese aspecto.

- Social Network Analysis with Content and Graphs [3].

Este estudio realiza una explicación sobre clusterización y creación de una base datos para la utilización de dicha técnica, etc.

Pros: -
Contras: Dado que el estudio actual no se debe de realizar por dicho método no es útil en este caso.

- A Graph Matching Method for HistoricalCensus Household Linkage [4].

El modelo propuesto no solamente considera la similitud del registro, sino que también incorpora la estructura del hogar. Según los datos se observa que no solo se puede obtener un incremento de la confiabilidad sobre sobre la vinculación al realizar vinculaciones grupales, sino que también, se podría realizar esta vinculación a otros registros grupales.

Pros: Mejora en la confiabilidad de las vinculaciones, posibles nuevos vínculos dados los registros grupales, un algoritmo de similitud de registros.
Contras: El algoritmo base de vinculaciones es poco adaptativo y se centra mayormente en la agrupación de registros por vivienda.

- Record Linkage: Current Practice and Future Directions [5].

Se explica y detalla un algoritmo básico y como conseguir estadísticas según la clasificación y los porcentajes obtenidos.

Pros: Algoritmo base, explicación de cómo conseguir ratios, variantes de algoritmo base con pros y contras según lo requerido, etc.
Contras: Algoritmo base sin abarcar muchas casuísticas, por lo que se deberán realizar expansiones, para que mejore.

- Multiple Instance Learning for Group Record Linkage [6].

Estudio en el que se realiza un método que trata enlaces grupales como bolsas y enlaces de registros individuales como instancias.

Pros: Una manera interesante y diferente de tratar los registros para realizar las vinculaciones.
Contras: Algoritmo cerrado y especializado en una metodología distinta a la que se va a tratar, por lo que dicho estudio no nos resulta útil dicho estudio.

- Duplicate Record Detection: A Survey [7].

Indica maneras de detectar duplicados en la misma base de datos.

Se ha escogido el estudio [5] porque detalla una metodología de algoritmo básica de Record Linkage. Esta explicación de la metodología sirve para poder realizar una implementación, esto permite que se pueda utilizar para la realización en cualquier lenguaje de programación. Entre otras cosas también es muy util este estudio ya que detalla diferentes técnicas de realización de resultados, cálculos, etc que dependiendo de las necesidades.

Aunque el [5] no detalla como poder realizar conexiónes entre familiares, vinculaciones entre sexo femenino, etc. Permite realizar una base solida para poder después complementarla con las expansiones que se necesiten.

4 METODOLOGÍA Y DESARROLLO

El lenguaje utilizado ha sido Python [8] debido a que es un lenguaje donde contiene mucha diversidad de librerías, funciones y conexiones con otras posibles aplicaciones que podrían darse en un futuro.

Todo el conjunto de datos es tratado mediante la librería Pandas [9] la cual ofrece multiindexación y un tratamiento de grandes cantidades de datos muy eficiente comparado con otras (Numpy, clases con Python, etc).

La metodología utilizada en todo el proyecto ha sido la incremental [10], dado que lo primero a obtener era un algoritmo básico y a partir de este ir realizando itereaciones para ir logrando mejoras y expansiones sobre el base. La ventaja obtenida dada esta metodología es que por cada iteración se iba obteniendo un algoritmo más completo y mejorado, por lo que se podían ir optimizando partes que hasta que no se realizaban expansiones no daban un rendimiento óptimo.

Gracias a los subojetivos, investigaciones y pruebas se ha podido realizar un código limpio, bien definido, versátil, mantenible y siguiendo las normativas de codificación pylint [11] gracias al linter disponible en sublimetext.

De entre todas las librerias dedicadas a Record Linkage, se ha escogido la librería recordlinkage [12], puesto que es una librería sencilla, abierta, compatible con la librería Pandas [9] y además tiene un soporte activo por la comunidad en su repositorio de github [13].

El diagrama de flujo básico que se realiza para un algoritmo de RecordLinkage es el siguiente:

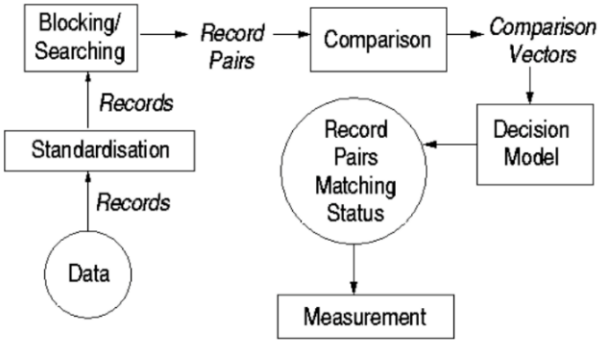


Fig. 1: Diagrama flujo básico. Imagen extraida de [5]

- *Data*: Se importa la información del Dataset.
- *Standardisation*: Estandarización y tratamiento de la información importada para poder ser utilizada.
- *Bloquing/Searching*: Selección de los registros posibles para el lincado utilizando atributos de bloqueo.
- *Comparison*: Se crean los vectores de comparaciones para poder obtener las distancias de los candidatos.
- *Decision Model*: Se ejecutan los vectores de comparación contra los candidatos y se obtienen los resultados de las comparaciones.
- *Measurement*: Obtenidos los candidatos finales se realizan las métricas para mostrar los resultados.

Dado un registro se realizará el *blocking and searching* de los posibles registros que pueden concordar, con este subconjunto se realizarán las comparaciones para estimar un porcentaje de exactitud, se comprobarán los estados de los candidatos generados y se realizarán las medidas.

$$\gamma = (\gamma^1(\alpha(a), \beta(b)), ..., \gamma^k(\alpha(a), \beta(b)))$$

Fig. 2: Formula de estimación de candidato

γ = % total estimado del registro.
γk = % estimado del registro dado un atributo.
α = Subconjunto del atributo K del primer censo
β = Subconjunto del atributo K del segundo censo

Los posibles registros se definirán según un % estimado por la función γ:

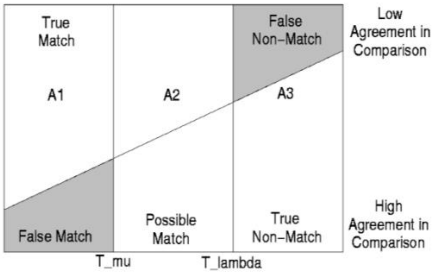


Fig. 3: Diagrama de estimación de candidato. Imagen extraida de [5]

- *match*, A1.
- possible *match*, A2.
- *non-match*, A3.

Todo el proceso de desarrollo se divide en diferentes secciones para tratar más adecuadamente todo el proceso realizado y sus casuísticas:

1. Main
2. Importación
3. Tratamiento de la información
4. Parámetros de usuario
5. Generación de candidatos
6. Creación/Ejecución de Discriminadores
7. Tratamiento de los resultados obtenidos
8. Presentación y exportación de resultados

4.1 Importación de Datos

Una de las partes más importantes de todo sistema es la correcta y rápida importación de datos. Este es un punto crítico en todo sistema ya que existen muchas codificaciones de datos y el programa debería comportarse ade-

cuadamente teniendo en cuenta dichos cambios. También se debe identificar cuál es el atributo que identifica de manera inequívoca a cada registro dentro de todo el conjunto de datos y los años de los censos para poder hacer una separación en el siguiente apartado.

4.2 Tratamiento de la información

Todos los datos importados siempre tienen "impurezas" en mayor o menor medida, para poder realizar un correcto procesamiento de estos se deben tratar. Gracias al trabajo realizado en la obtención de los datos ha habido pocos procesos a realizar para la limpieza de estos, exceptuando algunos índices repetidos y problemas en la codificación de datos, que se han podido tratar correctamente.

Dado que toda la información estaba mantenida en un solo archivo, se ha tenido que segmentar toda esta información según el parámetro de año de recogida de los datos 'any_padro'.

Todo este proceso tarda alrededor de 2 segundos para un tratamiento total de 59085 filas x 25 columnas.

4.3 Parámetros de usuario

Todo este proceso requiere de la intervención del usuario para poder adaptar la comparación entre censos de diferentes maneras (realizado mediante un archivo de configuración). El usuario podrá optar a la configuración de:

- Los 2 censos propuestos para el tratamiento.
- Peso de los campos comparados (0-100%) para darle más importancia o menos al resultado obtenido.
- El método de comparación para los campos de tipo texto, numéricos, coordenadas, fecha, etc.
- Porcentaje de acierto mínimo de los resultados para ser aceptados.

4.4 Generación de candidatos

Se utiliza una generación de posibles resultados de FULLxFULL, es decir, se realiza una combinación de todos los registros del Censo A con todos los registros del censo B. Dado que se tienen que explorar todas las posibilidades es necesario esta generación, la cual tiene un aspecto negativo, que es la obtención de una gran cantidad de candidatos y su tratamiento es algo costoso a nivel computacional.

4.5 Creación/Ejecución de Discriminadores

Para poder definir unas reglas de comparación entre los distintos atributos de los registros se pueden utilizar diferentes discriminadores [14]:

- **String**: Comparacion entre textos.
  - Jaro: Proporciona una medida de similitud entre dos cadenas que permiten transposiciones de caracteres.
  - Jaro-winkler: Proporciona una medida de similitud entre dos cadenas que permiten transposiciones de caracteres en un grado que ajusta la ponderación de los prefijos comunes.
  - Levenshtein: Diferencia entre dos secuencias, la distancia entre las dos palabras es el número mínimo de ediciones de un solo carácter (inserciones, eliminaciones o sustituciones) necesarias para cambiar una palabra por otra.
  - Damerau Levenshtein: Añade la opción de transposición al método de Levenshtein.
  - Q-gram: Suma de las diferencias absolutas entre los vectores q-gram de ambas cadenas. Se particiona la cadena en vectores de tamaño Q, con todas las posibilidades y se comprueba que las dos cadenas contengan estos vectores.
  - Cosine: 1 menos la similitud de coseno de ambos vectores Q-gram.
  - Smith Waterman: En lugar de mirar toda la secuencia, el algoritmo de Smith-Waterman compara segmentos de todas las longitudes posibles y optimiza la medida de similitud realizando una matriz según los resultados y haciendo al final un *traceback* con el resultado obtenido.
  - LCS: Busca la subsecuencia más larga común a todas las secuencias en el otro subconjunto de secuencias (a menudo solo 2 secuencias).
- **Exact**: Exactitud entre cualquier tipo de datos.
- **Geo**: Comparación entre puntos geométricos.
- **Date**: Distancia entre fechas.
- **Compare_vectorized**: Comparaciones realizadas por una función personalizada.
- **Numeric**: Comparaciones entre datos numéricos.
  - Step.
  - Linear.
  - Exp.
  - Gauss.
  - Squared.

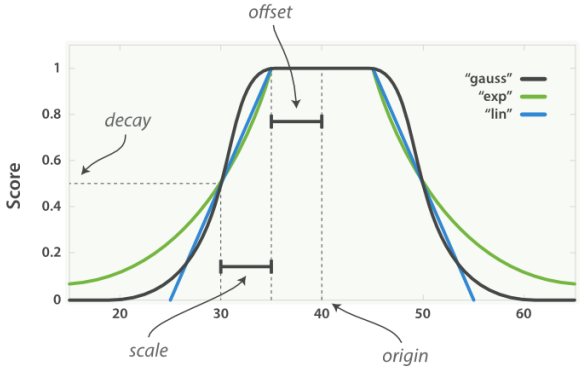


Fig. 4: Gráfico referente a la comparación numérica. Imagen extraida de [14]

La utilización de los anteriores discriminadores juntamente con las diferentes configuraciones de ellos es la 'clave' para poder obtener mejores resultados de las com-

paraciones entre registros y así conseguir unos resultados mejores.

Una vez se han definido todos los discriminadores se debe ejecutar la función que pone en marcha el proceso de comparación de resultados, dicha opción admite la utilización de paralelización porque cada discriminador es autónomo, se puede indicar con el atributo n_jobs.

4.6 Tratamiento de los resultados obtenidos

Los resultados obtenidos mediante los discriminadores están sin tratar y para obtener los emparejamientos definitivos es necesario realizar varios procesos:

- Multiplicar los valores de las diferentes distancias obtenidas de los discriminantes por el peso indicado por cada atributo.
- Ordenar los candidatos según la suma total de los resultados obtenidos.
- Eliminar aquellos candidatos que sean menores al valor mínimo requerido por el usuario.
- Recoger los resultados en orden descendente según la suma anterior de los valores y se van añadiendo a la lista definitiva de resultados, con la normativa de que un registro de A solamente puede estar con un registro de B y viceversa.

4.7 Presentación y exportación de resultados.

Para poder calcular los resultados de una manera sencilla, se ha realizado una obtención automática de los registros ya relacionados de los datos inicialmente importados, seleccionando solamente aquellos registros presentes en los censos indicados al inicio por el usuario.
Los emparejamientos finales se añaden en un fichero único por cada par de censos comparados. Se realiza un fichero conjunto con todas las comparaciones de censos, que contiene toda la información relevante a la ejecución de las comparaciones realizadas.

5 DATASET

El *dataset* proporcionado por el CED son censos y padrones que van desde el año 1828 al 1940 con una média de 8 años entre los diferentes censos, el formato es un fichero Excel, pero para mejorar la importación de todos los datos se traspasa a un fichero con extensión csv.

Todo el sistema está realizado para que pueda ser configurado fácilmente según distintos tipos de *datasets*, aun así, se debería intentar mantener el mismo formato para tener que realizar el menor número de configuraciones posibles.

Los campos traspasados en el *dataset* proporcionado son los siguientes:

- **id_general**: Identificador general (Año + Identificador individuo + Identificador hogar)
- **id_padro_individu**: Identificador de la persona a cada censo (Año + Identificador individuo)
- **id_padro_llar**: Identificador del hogar a cada censo (Año + Identificador hogar)
- **DNI:** Identificador de persona para poder seguirlo en el tiempo, de aquellos que han estado relacionados mínimo una vez.
- **any_padro**: Año del censo.
- **id_document**: Identificador de la imagen que se ha transcrito. Para poder recuperar la imagen física donde aparece el registro.
- **id_individu**: Indentificador individual. En cada censo hay un identificador de persona para que no se repita.
- **id_llar**: Identificador del hogar.
- **casa_carrer**: Nombre de la casa donde residen.
- **casa_num**: Número de la casa donde residen.
- **Noms_harmo**: Nombre harmonizado.
- **cognom1_harmo**: Apellido 1 harmonizado.
- **cognom2_harmo**: Apellido 2 harmonizado.
- **nom**: Nombre.
- **cognom_1**: Apellido 1 literal.
- **cognom_2**: Apellido 2 literal.
- **data_naix**: Data de nacimiento.
- **cohort**: Año de nacimiento.
- **edat**: Edad.
- **estat_civil**: Estado civil.
- **parentesc**: Relación de parentesco literal.
- **parentesc_har**: Relación de parentesco harmonizado.
- **ocupacio**: Ocupación literal.
- **nat_lloc**: Lugar de nacimiento.
- **nat_prov**: Provincia de nacimiento.
- **res_lloc**: Municipio de residencia.
- **res_prov**: Provincia de residencia.
- **temp_res_a**: Tiempo de residencia (años).
- **temp_res_m**: Tiempo de residencia en (meses).
- **ocupacio_hisco**: Ocupación harmonizada según la clasificación internacional HISCO.

Los campos utilizados por su mayor valor han sido:
nom, cognom_1, cognom_2, Noms_harmo, cognom1_harmo, cognom2_harmo, casa_carrer, casa_num, data_naix, cohort, edat, nat_lloc, nat_prov, estat_civil, ocupacio y ocupacio_hisco.

El resto de los campos, al ser de utilidad de identificación y/o tener información poca valiosa para la realización de ponderación, no se han utilizado.

6 IMPLEMENTACIÓN

A la hora de implementar el algoritmo se han realizado mejoras sobre distintas partes, ya que no eran del todo eficientes para la utilización en un ámbito real y para el usuario.

En este apartado se resumen los problemas encontrados y las soluciones implementadas más destacables que han surgido durante todo el proceso.

6.1 Importación de datos

Para conseguir una rápida, eficiente y neutra importación de datos, se han encontrado varios problemas porque no hay una única manera de hacerlo y se debe de adaptar a la utilización.
Se han realizado 3 tipos distintos de importación:

- Excel [15]
- Csv mediante librería csv [16]
- Csv mediante librería pandas [17]

En el primer caso se obtenía un tiempo de importación muy alto, además de haber adaptado la librería para una mayor rapidez.
En el segundo caso se obtuvo una rapidez más adecuada a la pensada, pero aun así existían problemas debido a los tipos de datos, ya que en Mac, Windows o Linux no son demasiado compatibles y no utilizan los mismos tipos de codificación, además de tener que traspasar todos los datos. Los tipos de datos utilizados por la librería RecordLinkage este proceso es algo delicado, lleno de trasnformaciones y lento.
Dados todos los problemas que causaba la importación de datos se decidió dar un enfoque más eficaz y se llevó la importación directamente con la librería pandas que se tuvo que parametrizar debido a que es una gran cantidad de datos y por defecto pandas no permite una importación tan elevada por el uso de memoria RAM.
Una vez los tiempos de importación eran aceptables, se hicieron pruebas sobre como solucionar la codificación de los datos. Después de varias pruebas y complejas codificaciones se optó por traspasar todo el programa a python 3.7 [18] donde se solucionaron todos los problemas de codificación.

Método de importación	Tiempo
Excel	Mayor a 60 minutos
Csv mediante librería csv	Aprox 6 segundos
Csv mediante librería pandas	1 segundo

Fig. 5: Tabla tiempo de importaciones

6.2 Separación de censos

La separación de años se realiza de manera dinámica, creando un diccionario donde las keys son los diferentes valores del campo "any_padro" y los valores las arrays de valores recogidos de los censos.

Este proceso tarda menos de 1 segundo.

6.3 Proceso de compraración

Todo el proceso de comparación, como ya se ha explicado, se realiza por atributos (columnas en el excel) independientes, pero hoy en día se utilizan procesadores de varios núcleos que son capaces de soportar varios hilos de ejecución simultánea, se estuvieron realizando unas pruebas dada la regla general:

$$2 * N^{o} \text{ Cores} - 1$$

El número de hilos que mejor redimiento global ha dado ha sido 4, este número depende de dónde se ejecute el algoritmo por lo cual se debería cambiar para poder adecuarlo a las diferentes situaciones.

6.4 Tratamiento de resultados

Al tener una gran cantidad de candidatos y tener que realizar operaciones sobre todos ellos para aplicar las ponderaciones de pesos, sumatorios y ordenación según el resultado cualificativo final, se han encontrado complicaciones ya que el tiempo del algoritmo aumentaba en dependiendo de los algoritmos de ordenación y del método utilizado al realizar las operaciónes aritméticas. Dado que se ha aplicado un buen tiempo sobre la ordenación de candidatos se ha disminuido mucho por ejemplo la operación de eliminación de candidatos donde el sumatorio de las compraraciones no llega a lo requerido por el usuario, dando así un tiempo de todo este proceso de 3 segundos para un total de aproximadamente 35 millones de candidatos

6.5 Gestión de resultados finales

Dado que únicamente puede existir un único registro del censo A relacionado con un único registro del censo B, no podrá existir otra relación entre los dos censos que incluyan cualquiera de los registros ya incluidos como relacionados, además la agregación de los *lincados* debe ser en orden descendente según el porcentaje total de acierto ponderado calculado anteriormente.

Este ha sido el mayor problema a nivel computacional de todo el programa, ya que no daba un resultado de tiempo aceptable para el uso diario. Se plantearon varias maneras para disminuir el tiempo crítico del algoritmo:

- Bucle de recorrido de todos los candidatos, dado que todos los candidatos están ordenados según el

porcentaje ponderado de puntuación, se hizo que por cada registro recorrido se eliminasen los demás candidatos que contengan cualquier registro de A o de B, dando así como lincado final el primer candidato encontrado. Se descartó este tipo de recorrido porque realizaba demasiadas búsquedas y eliminaciones, dando un tiempo de ejecución muy elevado.

- Se hizo un cambio para la utilización de recorrido del bucle para que se realizara con librerías numpy. Daba bastantes fallos en cuestión de conversiones, por lo que se descartó debido a que el planteamiento de mantener el mismo sistema de datos durante todo el algoritmo daba mejor resultado. Aun así, el tiempo crítico del programa bajó un poco.
- Bucle de recorrido de todos los candidatos, cuando se encuentra una relación final, se agrega en la lista definitiva y se borran todos los candidatos que ya no son válidos sobre el multiframe que se recorre. Esto da origen a un tiempo menor y mejor para pocos candidatos, pero muy elevado cuando existen muchos candidatos, debido a comparaciones, inserciones, borrados…
- El último método utilizado y actual, es realizar el método anterior con una partición del primer 25% de los candidatos dando casi finalizada la elección de lincados finales. Una vez terminado el recorrido de este 25% se elimina del 75% restante los que contienen registros de los licandos finales y se termina de recorrer los restantes. Este método tiene la particularidad de mejora de tiempo como mayor sea el número de candidatos.

Método	Tiempo(minutos)
Bucle con eliminaciones	90
Bucle con numpy	85
Bucle con lista	50
Bucle de 25% y 75%	12

Fig. 6: Tabla de tiempos de selección de candidatos

La tabla de tiempos ha sido realizada con un tamaño de candidatos de 33 millones.

La realización de este tipo de recorrido ha sido en relación a tiempo la mayor mejora del algoritmo porque se ha detectado y tratado el tiempo crítico dando una reducción muy grande y la posibilidad de la utilización del programa en un tiempo realista.

6.6 Automatización del programa

Una vez realizado y terminado todo el programa se tuvo que adaptar y estudiar una manera de que el usuario pudiera configurar los parámetros del algoritmo para el tratamiento de datos de una manera fácil y rápida, por lo que se decidió que la entrada de datos del algoritmo se realizara mediante archivos. El tipo de archivo elegido ha sido de extrensión csv, porque admiten una gran cantidad de datos fácilmente tratables y configurables.

7 Resultados

En esta sección se realiza un estudio de todos los datos resultantes del programa, todo el apartado de métricas, la selección de los parámetros base del algoritmo y los resultados finales obtenidos con dichos parámetros en los diferentes censos.

7.1 Métricas

Para poder tener una mejor comprensión de los resultados, se ha realizado una obtención automática de los registros ya relacionados de los datos inicialmente importados seleccionando únicamente aquellos registros presentes en los censos indicados por el usuario.

Gracias al punto anterior se pueden obtener los valores de la matriz de confusión:

- **True Positives**: Relaciones que se han detectado como aciertos.
- **True Negatives**: Relaciones que se han detectado como falsos.
- **False Positives**: Relaciones que se proponen como aciertos, pero no lo son.
- **False Negatives**: Relaciones que se han detectado como falsos pero que deberían ser aciertos.

Es importante tener en cuenta que los TP y TN deben ser muy altos y los FN deben ser muy bajos. Por otro lado, los FP deben tener un valor bajo ya que no interesa tener una gran cantidad de posibles relaciones, dado que se busca obtener *links* de gran calidad.

Con los valores anteriormente calculados se obtienen:

- **Precisión:** Porcentaje de resultados relevantes. TP/(TP+FP).
- **Recall:** Porcentaje de resultados relevantes totales clasificados correctamente. TP/(TP+FN).
- **Accuracy:** Porcentaje de aciertos tanto positivos como negativos. (TP+TN)/(TP+FP+TN+FN).
- **Specificity:** Porcentaje de acierto negativos. TN/(FP+TN).
- **Fscore**: Compraración entre precision y recall. 2*(precision*recall) / (precision+recall).

Los resultados obtenidos son exportados a un csv para la utilización de estos en softwares externos.

7.2 Selección de parámetros

Para poder selecciónar unos parámetros por defecto en el algoritmo se ha realizado un conjunto de pruebas con diferentes parámetros admitidos por el programa y se

han llevado a cabo todas las pruebas sobre un mismo censo para poder evaluar los resultados obtenidos.

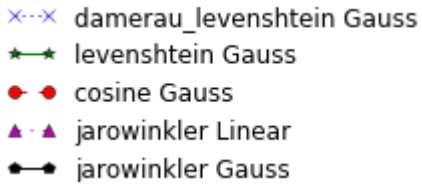


Fig. 7: Leyenda tablas selección parámetros

La leyenda no ha sido incorporada a los gráficos debido al poco expacio existente en ellos dada toda la información que se necesita visualizar.
Se han realizado las pruebas con los 4 algoritmos más usuales en compraraciones de texto y los métodos de compraración numérica que más se adaptaban al conjunto, solamente se han utilizado Linea y Gauss dado que este último era el más adiente para las comparaciones.

Los conjuntos de prueba se dividen en 10, se han divido en 10 grupos escogiendo los parámetros más informativos del *dataset*, donde se han utilizado los diferentes atributos:

- **A**: nombres y apellidos harmonizados.
- **B**: nombres y apellidos sin harmonizar.
- **C**: nombres y apellidos sin harmonizar, casa_carrer, casa_num, edat, ocupacio.
- **D**: nombres y apellidos sin harmonizar, casa_carrer, edat, ocupacio.
- **E**: nombres y apellidos sin harmonizar, casa_num, edat, ocupacio.
- **F**: nombres y apellidos sin harmonizar, ocupacio_hisco.
- **G**: nombres y apellidos sin harmonizar, edat, ocupacio, ocupacio_hisco.
- **H**: nombres y apellidos sin harmonizar, casa_carrer, casa_num, edat, ocupacio, ocupacio_hisco.
- **I**: nombres y apellidos sin harmonizar, casa_carrer, edat, ocupacio, ocupacio_hisco.
- **J**: nombres y apellidos sin harmonizar, casa_num, edat, ocupacio, ocupacio_hisco.

En la Figura (8), se puede observar que el tiempo es muy dependiente del algoritmo utilizado para la comparación de textos, dando una separación entre regiones muy amplia. Esto es debido a que la mayoría de los atributos son de tipo texto.

El algoritmo que mejor porcentaje de aciertos da es Jarowinkler, pero donde más cerca están es cuando se utilizan los conjuntos de pruebas E y G. Esta información se puede apreciar en la Figura (9) que indica el Recall.

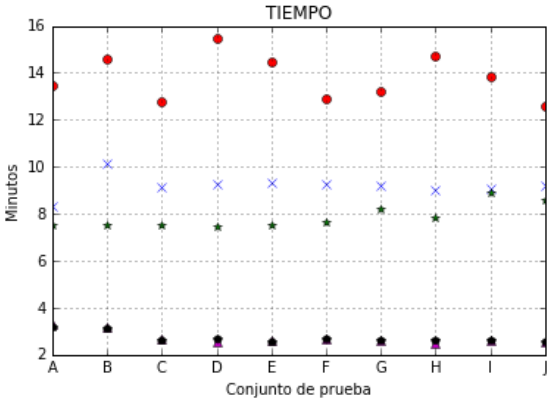


Fig. 8: Tabla de tiempos selección de candidatos

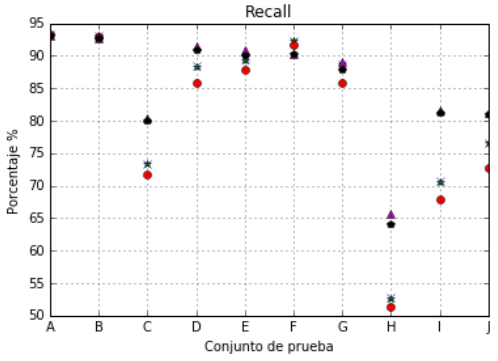


Fig. 9: Tabla recall de selección de candidatos

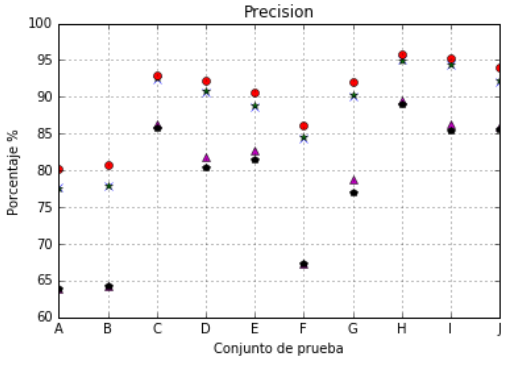


Fig. 10: Tabla precision de selección de candidatos

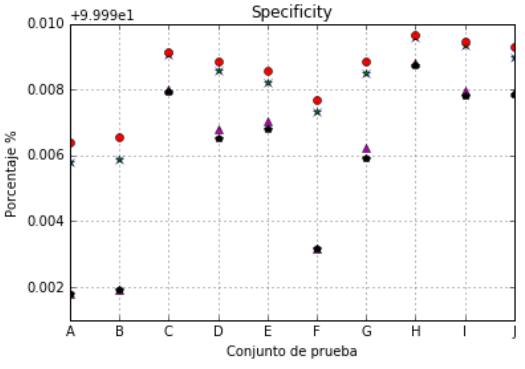


Fig. 11: Tabla specificity de selección de candidatos

Dadas las Figuras (9) y (10) se aprecia que los algoritmos de Levenshtein y cosine son los que detectan más TN

y TP, cuando existe mucho ruido en las variables de tipo texto, pero tiene la desventaja de que el coste computacional es mucho mayor y por lo tanto el tiempo necesario para la obtención de resultados es superior.

En los anexos 11.1 se pueden observar todas las tablas de precision y recall.

Analizados los datos obtenidos las variables por defecto en el algoritmo serán:

- Algoritmo alfanumérico: Cosine.
- Algoritmo numérico: Gauss.
- Atributos: pertinentes al conjunto G

7.3 Resultados

Marcados los parámetros por defecto, se han obtenido los resultados lanzando el programa con algunos de los censos traspasados por el CED. Dichos resultados son un poco ambiguos ya que hay muchas relaciones realizadas a mano por personas que tienen un conocimiento mucho más amplio y que no se han traspasado al programa porque se necesitaría un desarrollo más largo y especializado según la zona, años…

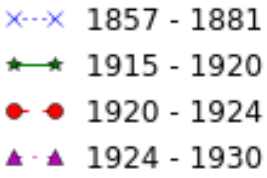


Fig. 12: Leyenda tablas resultados

Los tiempos como se pueden observar en la Figura (13) son parecidos a los de la Figura (8) en la línea de tiempo de Cosine. Cuanto más nuevo es el censo y menor es el peso mínimo, mayor es el tiempo ya que existen más candidatos a tratar.

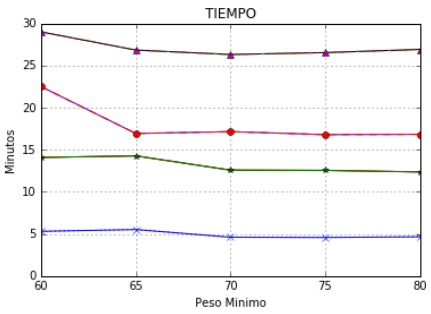


Fig. 13: Tabla de tiempos de resultados

Como se puede observar en las figuras (14) (15) (17) cada vez que se aumenta el peso mínimo se obtienen unos resultados mejores, hasta un punto que no son relevantes. Obtener una Precisión tan alta, como se obtiene apartir del peso 75-80 no es muy concluyente debido a que se obtienen muy pocos resultados, pero muy fiables y lo que se requiere, es la obtención de resultados fiables, pero con un cierto margen. Esta misma observación se puede ver en las demás figuras, en la figura (15) cada vez se obtienen menos resultados relevantes y en la figura (17) se obtinenen cada vez mas *true negatives*. En la figura (16) aparece claramente la curva carácterística de precision/recall, la cual indica que cuando aumenta la precisión disminuyen los resultados propuestos.

Los parámetros escogidos por el usuario influyen mucho en los resultados que se obtienen, por lo tanto, es muy importante escoger unos parámetros que seán lo más optimo posible al equipo donde se realiza la ejecución del programa y recordar que siempre es mejor optar por un peso mínimo no demasiado alto para poder obtener unos resultados más concluyentes.

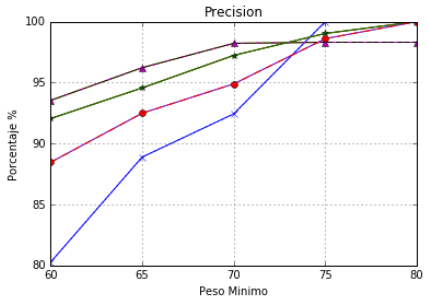


Fig. 14: Tabla Precision resultados

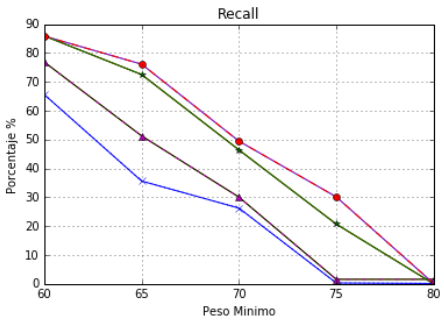


Fig. 15: Tabla Recall resultados

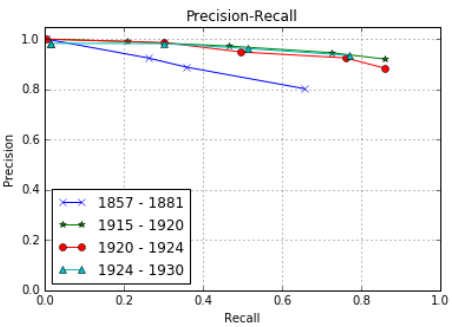


Fig. 16: Tabla Precision/Recall

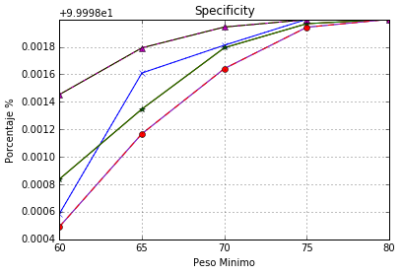


Fig. 17: Tabla Specificity resultados

Según las necesidades requeridas se podría escoger el algoritmo de jarowinkler ya que es algoritmo más rápido, pero algo menos eficaz, aunque con la configuración correcta es posible llegar a obtener unos resultados fiables y similares al algoritmo de Cosine, según los datos recopilados.

8 CONCLUSIONES

En este trabajo, se han llevado a cabo varios estudios sobre tratamientos de datos, implementación de un algoritmo RecordLinkage, importación de datos y mejoras en tiempos para permitir un uso real del programa realizado.

Dado que el programa tiene muchas posibilidades de configuraciones, se ha optado por realizar las más básicas, con las variables más utilizadas. Sería una buena mejora para el futuro realizar un conjunto de pruebas mucho mayor para poder analizar y recoger una configuración por defecto más eficaz que la empleada, ya que por limitaciones de tiempo no se han podido efectuar.

Aunque no se ha podido abarcar todo lo necesario para ser una herramienta definitiva para el CED, en el uso de relacionar personas en censos demográficos, sí que constituye una base muy importante para seguir ampliando según se vayan completando y mejorando los estudios realizados por los propios integrantes del CED y/o empresas colaboradoras, ya que cuanta más información harmonizada/estandarizada se obtenga, mejores resultados se obtendrán con el programa.

Durante el desarrollo del proyecto he aprendido muchos conceptos que desconocía (Recordlinkage, Pandas…), además de haber mejorado considerablemente en el manejo de tecnologías ya conocidas.

9 AGRADECIMIENTOS

En primer lugar, agradecer a Alicia Fornés por tutorizar este proyecto, ya que ha sido de gran ayuda para el entendimiento de todo el proceso.

También dar las gracias al equipo de CED por facilitarme todos los datos para la realización de pruebas y su gran trabajo a la hora de realizar los enlaces manuales.

Por último, agradecer a los compañeros que me han acompañado durante todos estos años, tanto dentro como fuera de la universidad.

10 BIBLIOGRAFIA

[1] CED. https://ced.uab.cat/

[2] Feigenbaum, James J. *A Machine Learning Approach to Census Record Linking*. In Retrieved March 28, pages 1-34, 2016.

[3] WM Campbell, CK Dagli, CJ Weinstein. *Social Network Analysis with Content and Graphs*. In Lincolnm Laboratory Journal, pages 1-20, 2013

[4] Fu, Zhichun and Christen, Peter and Zhou, Jun. *A graph matching method for historical census household linkage*. In Pacific-Asia Conference on Knowledge Discovery and Data Mining, pages 485-496, 2014.

[5] Gu, Lifang and Baxter, Rohan and Vickers, Deanne and Rainsford, Chris. *Record linkage: Current practice and future directions.* In CSIRO Mathematical and Information Sciences Technical Report, pages 83, 2003.

[6] Fu, Zhichun and Zhou, Jun and Christen, Peter and Boot, Mac. *Multiple instance learning for group record linkage.* In Pacific-Asia Conference on Knowledge Discovery and Data Mining, pages 171-182, 2012.

[7] Elmagarmid, Ahmed K and Ipeirotis, Panagiotis G and Verykios, Vassilios S. *Duplicate record detection: A survey* . In IEEE Transactions on knowledge and data engineering, pages 1-16, 2007.

[8] Lenguaje de programación Python. https://www.python.org/

[9] Librería Pandas. https://pandas.pydata.org

[10] Metodología incremental http://isw-udistrital.blogspot.com/2012/09/ingenieria-de-software-i.html

[11] Normativas Python: pylint. https://www.pylint.org

[12] Librería recordlinkage. https://recordlinkage.readthedocs.io/en/latest/about.html

[13] Github librería Recordlinkage. https://github.com/J535D165/recordlinkage

[14] Discriminadores record linkage. https://recordlinkage.readthedocs.io/en/latest/ref-compare.html

[15] Librería Openpyxl. https://openpyxl.readthedocs.io/en/stable/

[16] Libería CSV. https://docs.python.org/2/library/csv.html

[17] Import csv with pandas. https://pandas.pydata.org/pandas-docs/stable/generated/pandas.read_csv.html

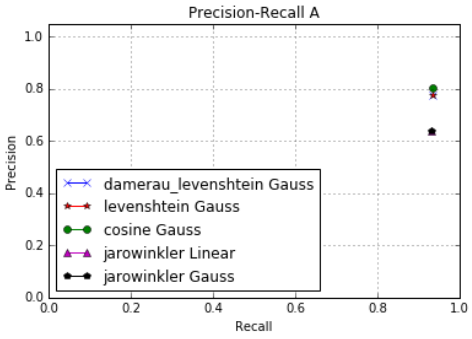
[18] Python 3.7. https://docs.python.org/3

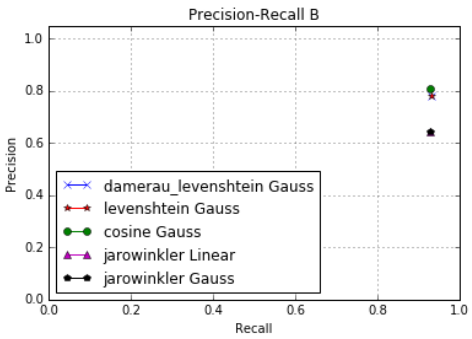
[19] Wikipedia record linkage. https://en.wikipedia.org/wiki/Record_linkage

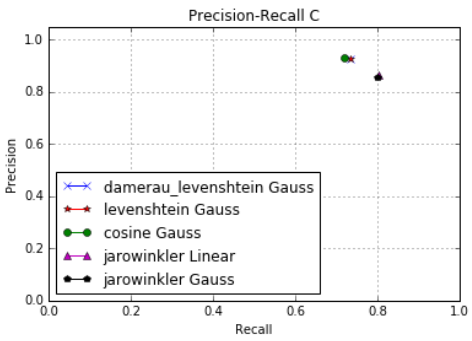
[20] Record linkage Redes. http://dag.cvc.uab.es/xarxes/info/

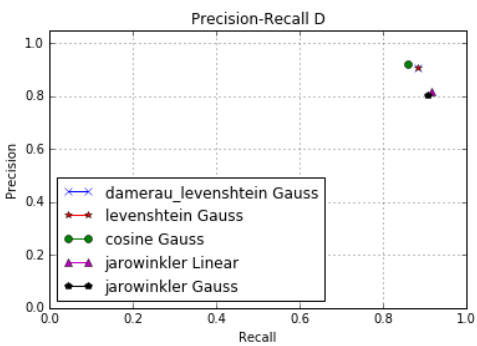
11 ANEXOS

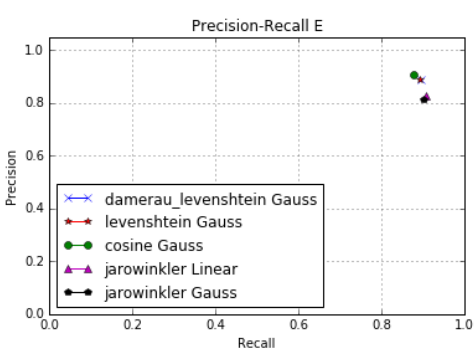
11.1 Tablas de precision recall selección de parámetros

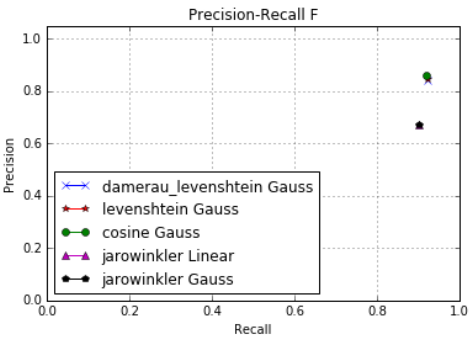


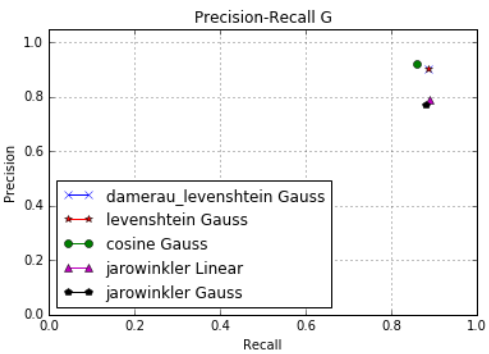


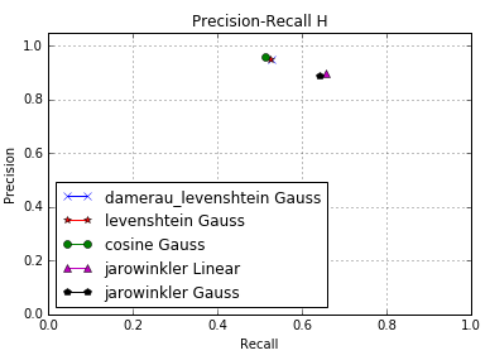


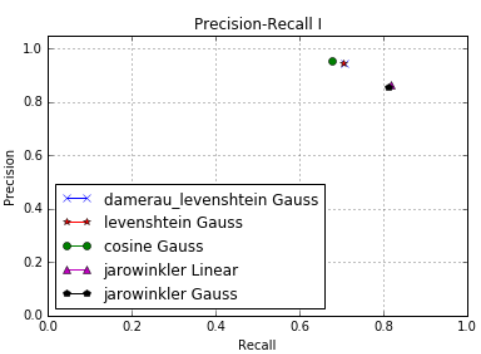


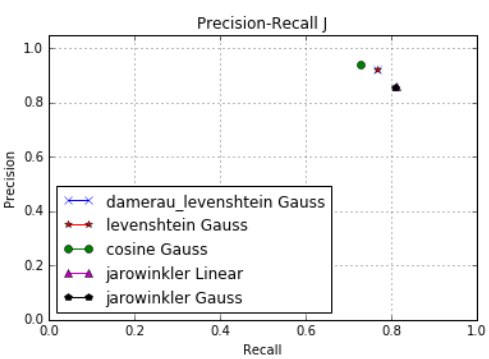




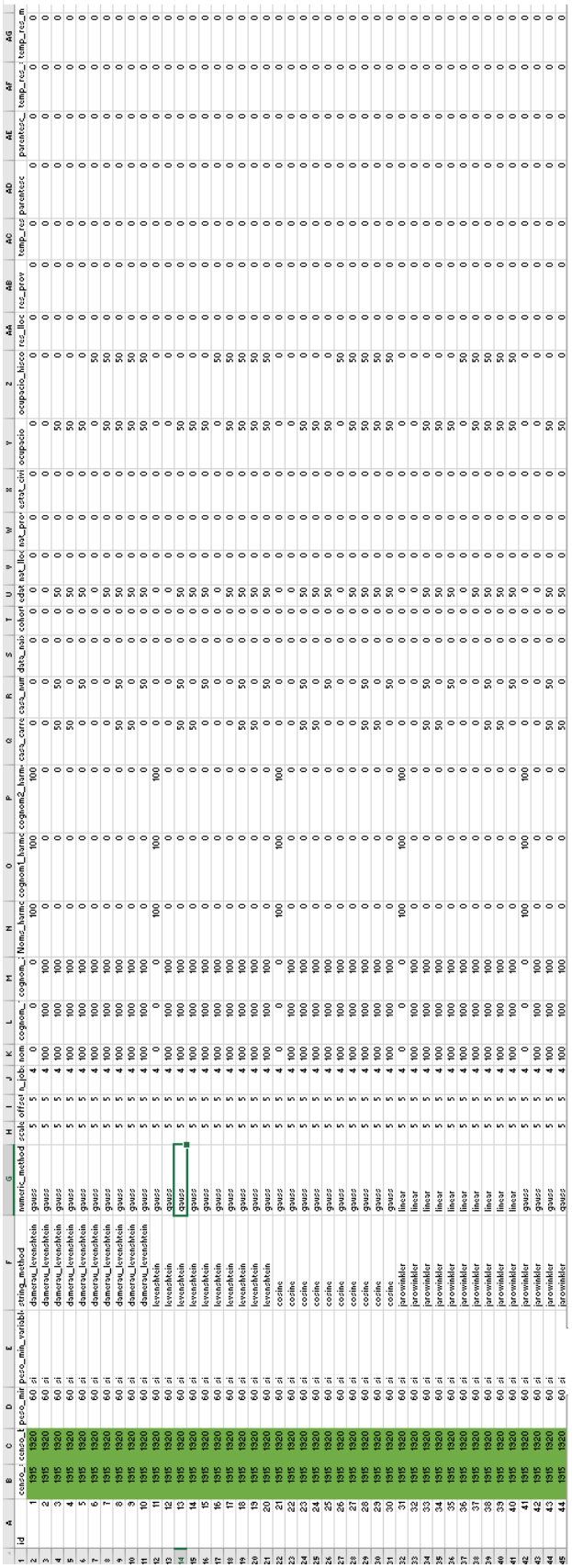








11.2 Ejemplo de parámetros del programa



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	AA	AB	AC	AD	AE	AF	AG
1	id	canso_	canso_t	peso_mir	peso_min_variable	string_method	numeric_method	scale	offset	n_jobs	nom	cognom_	cognom_:	Noms_harmc	cognom1_harmc	cognom2_harmc	casa_carre	casa_carre	casa_num	data_nais	cohort	edat	nat_lloc	nat_prov	estat_civi	ocupacio	ocupacio_hisco	res_lloc	res_prov	temp_res	parentesc	temp_res_	temp_res_m
2	1	1915	1920	60	si	damerau_levenshtein	gauss	5	5	4	0	0	0	0	100	100	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	2	1915	1920	60	si	damerau_levenshtein	gauss	5	5	4	100	100	100	100	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	3	1915	1920	60	si	damerau_levenshtein	gauss	5	5	4	100	100	100	100	0	0	0	50	50	0	0	50	0	0	0	50	0	0	0	0	0	0	0
5	4	1915	1920	60	si	damerau_levenshtein	gauss	5	5	4	100	100	100	100	0	0	0	50	0	0	0	50	0	0	0	50	0	0	0	0	0	0	0
6	5	1915	1920	60	si	damerau_levenshtein	gauss	5	5	4	100	100	100	100	0	0	0	0	50	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	6	1915	1920	60	si	damerau_levenshtein	gauss	5	5	4	100	100	100	100	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	7	1915	1920	60	si	damerau_levenshtein	gauss	5	5	4	100	100	100	100	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
9	8	1915	1920	60	si	damerau_levenshtein	gauss	5	5	4	100	100	100	100	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	9	1915	1920	60	si	damerau_levenshtein	gauss	5	5	4	100	100	100	100	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
11	10	1915	1920	60	si	damerau_levenshtein	gauss	5	5	4	100	100	100	100	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
12	11	1915	1920	60	si	levenshtein	gauss	5	5	4	100	0	0	100	0	100	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
13	12	1915	1920	60	si	levenshtein	gauss	5	5	4	100	100	100	100	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
14	13	1915	1920	60	si	levenshtein	gauss	5	5	4	100	100	100	100	0	0	0	50	50	0	0	50	0	0	0	50	0	0	0	0	0	0	0	0
15	14	1915	1920	60	si	levenshtein	gauss	5	5	4	100	100	100	100	0	0	0	50	0	0	0	50	0	0	0	50	0	0	0	0	0	0	0	0
16	15	1915	1920	60	si	levenshtein	gauss	5	5	4	100	100	100	100	0	0	0	0	50	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
17	16	1915	1920	60	si	levenshtein	gauss	5	5	4	100	100	100	100	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
18	17	1915	1920	60	si	levenshtein	gauss	5	5	4	100	100	100	100	0	0	0	0	0	0	0	0	0	0	0	50	0	0	0	0	0	0	0	0
19	18	1915	1920	60	si	levenshtein	gauss	5	5	4	100	100	100	100	0	0	0	50	50	0	0	50	0	0	0	50	0	0	0	0	0	0	0	0
20	19	1915	1920	60	si	levenshtein	gauss	5	5	4	100	100	100	100	0	0	0	50	0	0	0	50	0	0	0	50	0	0	0	0	0	0	0	0
21	20	1915	1920	60	si	levenshtein	gauss	5	5	4	100	100	100	100	0	0	0	50	50	0	0	50	0	0	0	50	0	0	0	0	0	0	0	0
22	21	1915	1920	60	si	cosine	gauss	5	5	4	0	0	0	100	0	100	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
23	22	1915	1920	60	si	cosine	gauss	5	5	4	100	100	100	100	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
24	23	1915	1920	60	si	cosine	gauss	5	5	4	100	100	100	100	0	0	0	50	50	0	0	50	0	0	0	50	0	0	0	0	0	0	0	0
25	24	1915	1920	60	si	cosine	gauss	5	5	4	100	100	100	100	0	0	0	50	0	0	0	50	0	0	0	50	0	0	0	0	0	0	0	0
26	25	1915	1920	60	si	cosine	gauss	5	5	4	100	100	100	100	0	0	0	0	50	0	0	50	0	0	0	50	0	0	0	0	0	0	0	0
27	26	1915	1920	60	si	cosine	gauss	5	5	4	100	100	100	100	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
28	27	1915	1920	60	si	cosine	gauss	5	5	4	100	100	100	100	0	0	0	50	0	0	0	50	0	0	0	50	0	0	0	0	0	0	0	0
29	28	1915	1920	60	si	cosine	gauss	5	5	4	100	100	100	100	0	0	0	50	50	0	0	50	0	0	0	50	0	0	0	0	0	0	0	0
30	29	1915	1920	60	si	cosine	gauss	5	5	4	100	100	100	100	0	0	0	50	0	0	0	50	0	0	0	50	0	0	0	0	0	0	0	0
31	30	1915	1920	60	si	cosine	gauss	5	5	4	100	100	100	100	0	0	0	0	50	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
32	31	1915	1920	60	si	jarowinkler	linear	5	5	4	0	0	0	100	0	100	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
33	32	1915	1920	60	si	jarowinkler	linear	5	5	4	100	100	100	100	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
34	33	1915	1920	60	si	jarowinkler	linear	5	5	4	100	100	100	100	0	0	0	50	50	0	0	50	0	0	0	50	0	0	0	0	0	0	0	0
35	34	1915	1920	60	si	jarowinkler	linear	5	5	4	100	100	100	100	0	0	0	0	50	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
36	35	1915	1920	60	si	jarowinkler	linear	5	5	4	100	100	100	100	0	0	0	0	0	0	0	0	0	0	0	50	0	0	0	0	0	0	0	0
37	36	1915	1920	60	si	jarowinkler	linear	5	5	4	100	100	100	100	0	0	0	50	0	0	0	50	0	0	0	50	0	0	0	0	0	0	0	0
38	37	1915	1920	60	si	jarowinkler	linear	5	5	4	100	100	100	100	0	0	0	0	50	0	0	50	0	0	0	50	0	0	0	0	0	0	0	0
39	38	1915	1920	60	si	jarowinkler	linear	5	5	4	100	100	100	100	0	0	0	50	50	0	0	50	0	0	0	50	0	0	0	0	0	0	0	0
40	39	1915	1920	60	si	jarowinkler	linear	5	5	4	100	100	100	100	0	0	0	0	50	0	0	50	0	0	0	50	0	0	0	0	0	0	0	0
41	40	1915	1920	60	si	jarowinkler	linear	5	5	4	100	100	100	100	0	0	0	0	50	0	0	50	0	0	0	50	0	0	0	0	0	0	0	0
42	41	1915	1920	60	si	jarowinkler	gauss	5	5	4	0	0	0	100	100	100	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
43	42	1915	1920	60	si	jarowinkler	gauss	5	5	4	100	100	100	100	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
44	43	1915	1920	60	si	jarowinkler	gauss	5	5	4	100	100	100	100	0	0	0	50	50	0	0	50	0	0	0	50	0	0	0	0	0	0	0	0
45	44	1915	1920	60	si	jarowinkler	gauss	5	5	4	100	100	100	100	0	0	0	0	50	0	0	50	0	0	0	50	0	0	0	0	0	0	0	0

11.3 Ejemplo de resultados

Las 3 primeras filas indican la información original del dataset y las 2 últimas el resultado optenido, se marcan las columnas con cambios.

id_general	nom	cognom_1	cognom_2	Noms_harmc	cognom1_harmc	cognom2_harmc	casa_carrer	casa_num	cohort	edat	nat_lloc	ocupacio_hisco	temp_res_a
19204190089	antonio	cahue	huguet	antonio	cahue	huguet	torrente	12	1898	22	esta	-102	22
19154140089	antonio	cauhe	hugnet	antonio	cahue	huguet	torrente	12	1898	17	esta	-102	17
id_general_1	id_general_2	nom	cognom_1	cognom_2	Noms_harmc	cognom1_harmc	cognom2_harmc	casa_carrer	casa_num	cohort	edat	nat_lloc	ocupacio_hisco	temp_res_a
1,91541E+11	1,92042E+11	0.14285714285	0.142857142857142	0.119047619047619	0.0	0.0	0.0	0.071428571428571	0.071428571	0.0	0.035714285	0.0	0.071428571428571	0.0