Aplicativo para la estandarización de nombres de veredas

Nicolás Ordóñez Chala
Estudiante de Ingeniería de Sistemas y Computación
Cel: +57 310 269 73 09
nordonezc@unal.edu.co

Universidad Nacional de Colombia
Facultad de Ingeniería
Bogotá, Colombia
2017

Contenido

1. Introducción

En Colombia, se están haciendo estudios sobre unas enfermedades, leptospirosis (todo el país) y leishmaniasis(Norte de Santander, por el momento), causadas por mosquitos. Por lo cual se están realizando unos reportes donde se ingresen datos geográficos de los pacientes que son atendidos con el fin de tener un control completo de cada uno y hacer un estudio pertinente. Para facilitar la tarea se requiere que los nombres de todos los lugares donde se realiza estén estandarizados, por lo cual se requiere una aplicación que ejecute esta tarea y automatice el proceso lo mejor posible.

2. Objetivo

- Automatizar el proceso de estandarización de nombres de los reportes de enfermedades de leptospirosis y leishmaniasis.
- Facilitar la generación de vectores para la ubicación de enfermedades.

3. Requerimientos

Entrada:

- Lectura de tablas para la lectura de información ingresada en reportes.
- Lectura de tablas donde se encuentran los códigos de los MUNICIPIO.
- Lectura de tablas donde se encuentran las tablas con los nombres de las veredas estandarizados.

Procedimiento:

- Analizar por fila todas las celdas donde se encuentran los nombres de las veredas.

Salida:

- Retorna una tabla donde se encuentren los códigos estandarizados que se ingresaron inicialmente.

4. Metodología para abordar el problema

Para la realización del problema anterior, se hará:

1. La lectura de una primera tabla, que será aquella que contiene la base de datos de todas las veredas y municipios.
2. Ingreso de todos los datos a una tabla Hash(La llamaremos TNE: tabla de nombres estandarizados).

3. Lectura de un segundo archivo que será donde se requiere llenar la tabla de nombres estandarizada.
    a. Lectura de las celdas donde pueda estar un posible nombre de la vereda.
    b. Eliminación de redundancia de palabras y caracteres cómo: "La", "vereda", "corregimiento", "de", "-", "/", "."
    c. Ingresar las palabras restantes a una arraylist (La llamaremos PE: Palabras por estandarizar).
4. Realizar búsqueda de similitud entre cada una de los contenidos de TNE y la PE, y retorna una tabla con las posibles nombres de las veredas, junto con un porcentaje o calificación de similitud.
    a. Comparación usando la métrica de la distancia de Levenshtein. Seleccionar aquel que tenga mayor porcentaje de similitud, y en caso de que no encuentre alguno, es decir, todos los valores son iguales al 0%, establecerlo como: No verificable.
5. Exportar dicha tabla a formato de excel.

5. Herramientas y algoritmos

Con recomendación del profesor de la Universidad de Santander y un poco de investigación posterior se encontró en github la implementación de un algoritmo que usa lógica difusa para comparar cadenas de caracteres retornando un valor numérico en términos de porcentaje de la siguiente manera: 0 no tiene ninguna semejanza y 100 tiene total semejanza. Dicho algoritmo utiliza la distancia de Levenstein para determinar el porcentaje de similitud entre dos palabras.

Aunque existen diversas técnicas de comparación usando programación dinámica, como lo es Needleman–Wunsch algorithm o Smith–Waterman algorithm, hasta el momento no se han encontrado estudios que planten una comparativa entre todos estos, y al todos usar la misma base, no parece que tengan muchas diferencias, e incluso existen herramientas como node.js que tienen proyectos como "Natural" que también retornan comparativas entre dos cadenas de caracteres.

En cuanto a la lectura de datos, python posee herramientas para la lectura de celdas en excel, lo que permite la transitividad de una herramienta a a otra para el análisis respectivo de cada uno de los caracteres. Finalmente, el resultado final de combinar estas herramientas será el proyecto inicialmente planteado utilizando cualquiera de los algoritmos junto con la escritura de datos apropiado y tratando de dejar una interfaz de usuario amigable para las personas destinadas a usar esta aplicación, que pueden ser desde

profesores, hasta estudiantes auxiliares que se contraten para realizar la labor pedida, o cualquier tarea que se asemeje a la del problema.

Se establece un repositorio en GitHub, para tener un control de las versiones ofrecidas por el estudiante hasta el momento. https://github.com/nordonezc/Standarapp-NVyCP/

6. Estimación de costos y tiempos

En base al artículo 2 de la resolución 000426 de 2014 de Colciencias, se establecen los pagos al estudiante investigador, teniendo las siguientes consideraciones:

- El pago se efectúa en base a las horas realizadas.
- El pago se calcula según la formación del estudiante (Título profesional, sugerido por el profesor).
- Se estima experiencia entre 1 y 5 años.

7. Anexos entregados

- Documento de excel con la tabla donde se debe llenar el nombre de la vereda con el nombre estandarizado.
- Documento de excel con el código de cada uno de los departamentos.
- Archivo donde se encuentran los nombres de todas las veredas con su respectivo municipio, para Santander.
- Archivo de enfermedades para la asignación de vectores
- Archivo para la generación una base de datos con centroides.

8. Problemas de lectura de archivos

Los archivos de excel entregados, tienen algunos problemas con caracteres especiales, para esto, se hizo el siguiente reemplazo(Se estandariza el tipo de letra en calibri 11):

- ┴ por A.
- ╔ por E.
- ═ por I.
- Ë por O.
- ┌ por U.
- ╠ por I.
- Ê por O.
- ▄ por U.
- ß por a.

- Ú por e.
- Ý por i.
- ¾ por o.
- · por u.
- š por u. (U con diéresis)
- ├âÔÇÿ por N, que sería una Ñ.
- ├â┬ü por A.
- ├âÔÇ░ por E.
- ├â┬ì por I.
- ├âÔÇ£ por O.
- ├â┼í por U.
- Ð por N.
- ± por N.

9. Aportes reuniones virtuales

Desde el principio del presente proyecto, se realizaron reuniones virtuales gracias a la herramienta Skype, donde por lo general, se hace una charla de voz y se comparten escritorios para entender el punto de vista del otro. todo lo anterior para facilitar la comunicación con el profesor Juan David Gutierrez. A continuación se presentan las reuniones realizadas y lo acordado en esa fecha:

- **4 de Noviembre de 2016(19/11/16):** Se hizo el planteamiento formal del problema, descrito en el numeral uno, y se dio un plazo de dos semanas para hacer un estudio previo sobre la viabilidad del proyecto, herramientas existentes y disponibilidad de tiempo. Profesor anexo varios documentos para tener un mayor acercamiento al problema.
- **19 de Noviembre de 2016(19/11/16):** Después del estudio realizado en las dos semanas acordadas, se mostró el presente documento y requerimientos algo más específicos en base a los resultados mostrados. Se agregaron nuevas tareas como el paso de formato .dbf a .xls; la eliminación de caracteres con tildes. Se acordaron los pagos por horas, y la fecha a partir de la cual se iba a empezar a trabajar formalmente en el proyecto.
- **30 de Noviembre de 2016(30/11/16):** La fecha inicial de trabajo, planteada inicialmente, comienza con una nueva reunión, donde se acuerda cambiar la estandarización de nombres a una estandarización de códigos y se recomienda enfocarse más en la exactitud del programa que en la optimización del tiempo del mismo. Se logra convertir de formato .dbf a .xls solo cambiando las

extensiones y por último, se realiza división de tareas donde el profesor se encarga de arreglar algunas celdas donde están los nombres de veredas y centros poblados.

- **9 de Diciembre de 2016(9/12/16):** Se pide revisar aquellas localidades que tengan un levenstein menor y no concuerdan con ninguna localidad en el municipio dado en el registro. dejar una anotación en caso de encontrar dicha localidad en un municipio diferente. Localidades si es menos de 80% descartado; municipios si es menos de 50% descartado. Aclarar codigo para arkgis. (respuesta al codigo: http://gis.stackexchange.com/questions/99615/writing-conditional-if-then-statements-into-field-calculator-of-arcgis-for-des).
- **14 de Diciembre de 2016(14/12/16):** Se pide leer registros en orden jerárquico: **1. procedencia 2. residencia** 3.**notificación**. Se encontró error en los registros de norte de santander, esto es suministrado directamente por el instituto nacional de Salud. Cuando se encuentre una dirección, se toma el municipio de residencia y se hace levenstein con la localidad(Tener en cuenta todos los posibles strings que pueden comenzar la direccion: #, N°, CL, K, KR, CLL, CALLE, CARRERA, Barrio, AV, Avenida). Revisar registro 1333 que soltaba nullpointerexception.
- **20 de Diciembre de 2016(20/12/16):** Se hacen múltiples correcciones en la lectura del archivo dbf debido a problemas por caracteres especiales generados por el cambio de formato de archivo. Adicional, se recuerda que las veredes que sean "Sin definir" no serán agregadas en la tabla donde se haga la búsqueda de los registros.
- **19 de Enero de 2016(19/01/16):** Se hace una videoconferencia con la profesora Ruth y el profesor reinaldo, junto con Juan David, donde se establece que a dirección solo se puede tener en cuenta si el municipio de residencia es igual al de procedencia.
- **19 de Enero de 2016(19/01/16):** Se determina la lectura correcta de los registros, por lo que se leerá el código del departamento y municipio, de la misma forma el algoritmo se seguirá aplicando para determinar el nombre correcto de la localidad, con la excepción de que solo se leerá la dirección si municipio de notificación y procedencia son iguales. (Ayuda brindada por el profesor Reinaldo y la profesora Ruth).
- **28 de Enero de 2016(28/01/16):** Se plantearon los parámetros de salida de lectura, y se propone la lectura de una nueva base de datos más completa.
- **Año 2017:** Surgió un nuevo requerimiento en el que se desea una nueva opción que genera los vectores para un archivo de excel dados el municipio, localidad y fecha.

10. Manual definitivo v3

Antes de ejecutar el archivo, revisar que exista una versión actualizada de java, revisar este link: http://www.oracle.com/technetwork/java/javase/downloads/jre8-downloads-2133155.html.

IMPORTANTE: Los archivos que lee la aplicación son solo .xlsx

Al ejecutar el archivo aparecerá una pantalla con cuatro opciones, de la siguiente manera:

- **Arreglar caracteres:** Opción creada para eliminar caracteres especiales un archivo de excel.

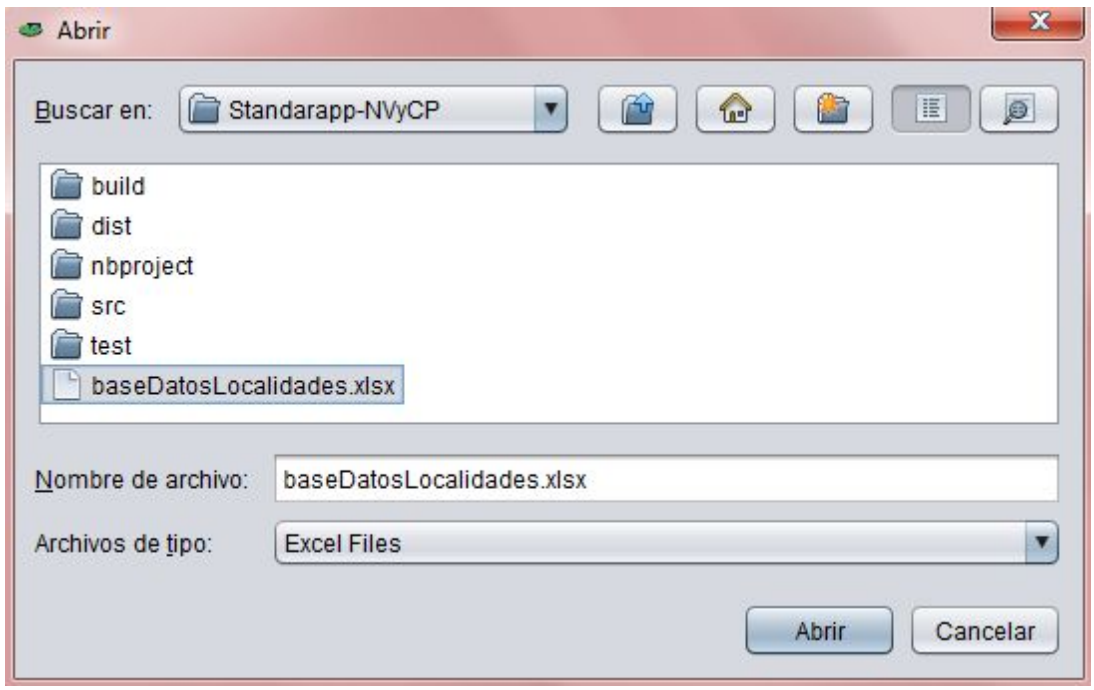
NOTA: Las opciones siguientes solicitarán que se seleccione el archivo baseDatosLocalidades.xlsx para poder funcionar.

- **Lectura de casos:** Creado con el fin de estandarizar los nombres escritos en la ficha de enfermedades.
- **Búsqueda personalizada:** En caso de ser necesario, se puede hacer búsquedas particulares para confirmar algún dato particular, donde se mostrará las coordenadas y el código
- **Muestro de vectores:** Se asignan los vectores de un registro dados unos datos particulares

Ahora, se dará una explicación más detallada de cada opción. Al seleccionar la primera opción, se mostrará la siguiente ventana:

En esta ventana, se solicita un archivo de entrada en formato .xlsx (hoja de cálculo de excel 2013). Igualmente, se solicitará una ubicación de salida, en caso de querer sobrescribir el archivo de entrada, se puede dejar está opción en blanco.

Para cuando se selecciona, cualquiera de las siguientes tres opciones (2, 3 y 4) se deberá seleccionar el archivo baseDatosLocalidad.xlsx, para que el programa funcione correctamente:



En caso de seleccionar la opción dos, aparecerá la siguiente ventana:

Se solicitará el archivo de la ficha de reporte, el cual tiene que estar en formato **.xlsx (EN CASO DE TENER OTRO FORMATO, NO SERVIRÁ),** se solicitan unos campos esenciales para el funcionamiento de la aplicación, cabe resaltar que viene con unos **NÚMEROS PREDETERMINADOS,** por lo que no deberían cambiarse, pero en caso de que exista un problema, preguntar al encargado de hacer la lectura de la ficha:

- ★ **Cod dpto ocurren** Corresponde al número de columna donde se encuentra el código del departamento de ocurrencia.
- ★ **Cod Mncp Ocurren:** Corresponde al número de columna donde se encuentra el código del municipio de ocurrencia.
- ★ **Localidad:** Corresponde al número de columna donde se ingresa el nombre de la localidad.
- ★ **Centro poblado:** Corresponde al número de columna donde se ingresa el nombre del centro poblado.
- ★ **Vereda :** Corresponde al número de columna donde se ingresa el nombre de la vereda.
- ★ **Barrio vereda:** Corresponde al número de columna donde se ingresa el nombre del barrio de la vereda.

- ★ **Dirección residencia:** Corresponde al número de columna donde se ingresa el nombre de la dirección de residencia.
- ★ **Municipio procedencia:** Corresponde al número de columna donde se ingresa el código del departamento de procedencia
- ★ **Municipio residencia:** Corresponde al número de columna donde se ingresa el nombre del municipio de residencia.
- ★ **Numero de pagina:** Corresponde al número de columna donde se ingresa la página del archivo de excel donde se encuentran los registros
- ★ **Porcentaje de comparacion:** Corresponde al número de columna donde se ingresa el porcentaje de similitud para determinar el nombre estandarizado, entre más bajo, más alto será el margen de error de la aplicación.

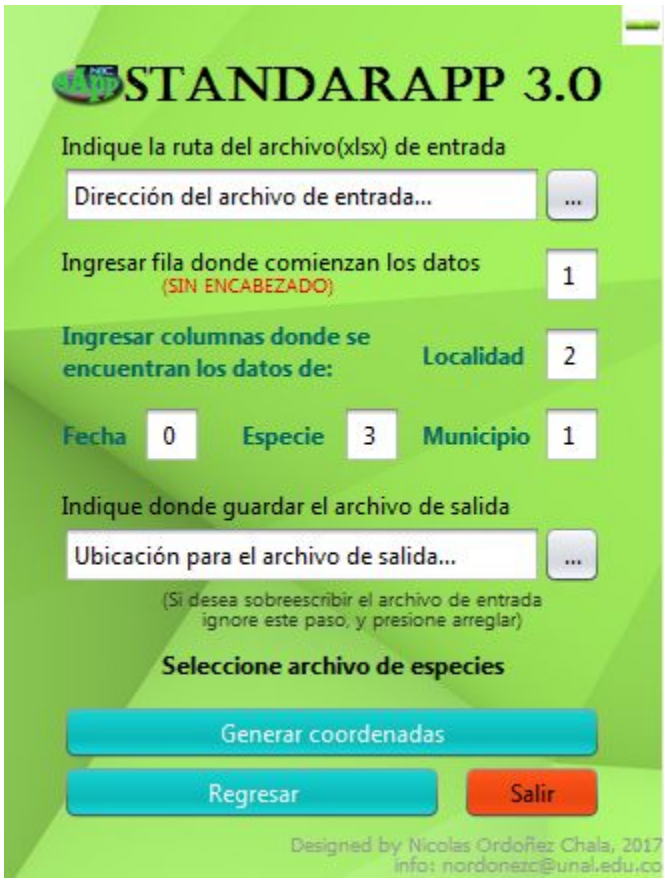
Finalmente, si se desea se puede seleccionar la ubicación donde se guardará el archivo con los nombres corregidos. Donde saldrá con el siguiente formato:

Cod_Dpto	Departame	Cod_Mncp	Municipio	Cod_Local	Localidad	X	Y	Levenstein
54	NORTE DE	54206	CONVENC	54206039	LA TRINIDAD	1087943	1450999	88
54	NORTE DE	54206	CONVENC	54206016	CULEBRITA	1078860	1426818	100

Para la siguiente opción, es en caso de que se desee realizar una búsqueda personalizada:

Aquí se puede realizar búsqueda personalizada de centros poblados, veredas y localidades con cierta flexibilidad, de manera que permita el ingreso de un código o un nombre y se realizará la búsqueda que más se parezca, si se desea hacer una búsqueda más exacta es preferible abrir el archivo baseDatosLocalidades.xlsx y hacer una búsqueda en excel. Una breve guía aqui: https://exceltotal.com/filtros-en-excel/

Para la última opción, se pedirá igualmente unos campos necesarios para hacer el correcto escrutinio de la información.



Se solicitará información semejante a la opción anterior, se exigirá el archivo que se le desea generar los centroides, y se pedirán los campos esenciales los cuales son:

- **Fila donde comienzan los datos:** Algunos archivos vienen con encabezado, esta fila será la última fila correspondiente a un encabezado, por lo que en caso de no tener encabezado se debe cambiar a cero, si el archivo de excel tiene un encabezado que termina hasta la fila número 3, se deberá ingresar 3 en este campo.
- **Localidad, especie, fecha, y municipio:** Es la columna correspondiente a dónde se encuentra cada uno de los datos mencionados.

Igualmente, se puede seleccionar ubicación de salida, pero está no será obligatoria.

11. Software considerado

Para el presente aplicativo, se ha hecho uso de varias herramientas y lenguajes de programación como:

- Quantum Gis/Gidahatari
- NetBeans IDE
- GitHub
- Java
- DBFcommander - TotalCommander

12. Fuentes de ayuda y bibliografía

- Implementación de comprador de cadenas usando lógica difusa en python con opción en java. https://github.com/seatgeek/fuzzywuzzy
- Foro SuperUser http://superuser.com/questions/437387/comparing-similar-text-strings-in-excel
- Levenshtein distance. Métrica de comparación. https://en.wikipedia.org/wiki/Levenshtein_distance
- QuenSu https://tekmarathon.com/2012/10/05/best-searching-algorithm-2/
- TekMarathon. https://tekmarathon.com/2012/10/05/best-searching-algorithm-2/
- Guia para lectura de datos en python de varias celdas de excel. https://www.sitepoint.com/using-python-parse-spreadsheet-data/
- Lectura y escritura de datos en excel desde python. http://stackoverflow.com/questions/19246844/reading-multiple-excel-files-in-a-directory-and-write-a-particular-cell-value-to
- Información sobre el grafo de Bruijin. https://en.wikipedia.org/wiki/De_Bruijn_graph
- Baeza-Yates R, Navarro G. "Fast Approximate String Matching in a Dictionary" (PDF). Proc. SPIRE'98. IEEE CS Press. pp. 14–22.
- Apache POI para la lectura de archivos .xls en java http://poi.apache.org/spreadsheet/index.html
- ArrayList contra LinkedList. http://www.enrique7mc.com/2016/07/diferencia-entre-arraylist-y-linkedlist/
- Posible repositorio donde se puede leer archivo dbf https://bitbucket.org/apuntesdejava/dbf2java-lib/src
- Más (+).