Facultad de Matemática y Ciencias de la Computación

Universidad de La Habana.

This Thesis Has no Name

Trabajo de Diploma presentado en opción al título de

Licenciado en Ciencia de la Computación

Autor: Elias Bestard Lorigados

e.bestard@estudiantes.matcom.uh.cu

Tutor: Lic. Pedro Quinteras Rojas

La Habana, 2019

Agradecimientos

Muchas Gracias ☺.

Resumen

En este trabajo se muestra las etapas de desarrollo, y los detalles de implementación de una aplicación con interfaz de comando de línea (*CLI* por sus siglas en ingles) para procesar y visualizar información Dando lugar a un programa con mucho campo.

Tabla de Contenidos

Tabla de Figuras

Introducción

Vivimos en un mundo repleto de información y diariamente se crean múltiples archivos, ya sean provenientes del resultado de una nueva investigación científica, una noticia, juegos de datos generados por un programa o algo tan simple como una tabla de datos con las estadísticas finales de los resultados del trabajo diario, así como de eventos de la vida cotidiana. Gran parte de esta información no se archiva necesariamente de la mejor forma posible, muchas veces se almacena sin estructura alguna que facilite su utilidad futura; e incluso, en casos, es generada y almacenada bien estructurada pero no de la manera adecuada para transmitir el mensaje correcto. De tal forma que en general en muchos casos que se requiera procesar información, ya sea para justificar trabajos, investigaciones, para analizar los resultados de cierta encuesta, transmitir cierto mensaje o simplemente para palpar los resultados obtenidos, es común ver que se hace de forma manual, y en este caso se pueden perder ciertas perspectivas e información útil.

El sistema visual de los seres humanos (eje o circuito ojo-cerebro) es el más sofisticado procesador de información jamás desarrollado, a través de graficas podemos poner en buen uso este sistema para obtener una visión más profunda de la estructura de los datos. Ejemplo de esto es lo sencillo que le resulta al ojo humano reconocer datos agrupados en gráficos de puntos (reconocer *clusters)*. Grandes cantidades de información cuantitativa puede converger en gráficos, donde nuestro sistema visual puede resumir vasta información rápidamente y extraer resultados, además que es capaz de enfocarse en pequeños detalles [1].

Los gráficos son representaciones visuales de información numérica o espacial y mayormente son más fáciles de leer que listas de números o tablas bien complejas. A continuación, vemos varias ventajas de graficar en el uso de estos:

1. Los gráficos pueden tener gran densidad de información, a veces sin perder datos.
2. Permiten rápida asimilación del resultado general.
3. Un mismo grafico puede ser visto de distintos niveles de detalle (ejemplo la impresión general, acercamiento y ubicación exacta de varios puntos adyacentes).
4. Pueden fácilmente mostrar complejas relaciones a través de varias variables (en dos, tres, o incluso más dimensiones).

Por estas razones, los gráficos son una parte importante de casi todo experimento o campo de base de tesis, investigación, artículo científico o presentación de una conferencia [2].

Existen ramas de las ciencias, tanto sociales como exactas que estudian como procesar y extraer información de documentos y algunas como mostrarlas de la mejor manera posible, ejemplo de estas son: el periodismo de datos, la visualización de datos, minería de datos (por

sus términos en ingles *data mining*), minería de textos (por sus términos en ingles *text mining*), etc. El enriquecimiento de estas ramas ha dado lugar al desarrollo de diversas herramientas, en muchos casos herramientas con fines específicos y con algunas limitantes.

La Facultad de Matemática y Ciencia de la Computación de la Universidad de la Habana se ha propuesto el desarrollo de una aplicación en este campo, contando con una herramienta de nuestra autoría que nos permitirá adaptarla a nuestras principales necesidades.

Por lo descrito anteriormente, surge la inquietud de responder el siguiente problema:

¿Será posible graficar información a partir de documentos de los cuales se pueda desconocer cómo está estructurado?

Objetivo general

Como objetivo de este trabajo nos planteamos desarrollar un software que permita graficar información a partir de documentos de los cuales se pueda desconocer cómo están estructurados.

Objetivos específicos

1. Estudiar herramientas actuales que permitan el desarrollo del *software*.
2. Proponer diseño modular que permita extraer información de cadenas de textos y graficarla.
3. Proponer un mecanismo de extracción de datos sencillo que permita reconocer patrones en cadenas de texto.
4. Proponer un mecanismo de aprendizaje sencillo que permita seleccionar de forma inteligente gráficos partiendo de un conjunto de datos.
5. Implementar un prototipo del modelo establecido con una interfaz de usuario amigable, (CLI).
6. Validar el prototipo presentado.

El presente documento está compuesto por los siguientes acápites. Estado del arte, que aborda una revisión bibliográfica sobre los temas de procesamiento, extracción de información, visualización de datos, selección de mejores gráficas; mostrando herramientas que tratan estos temas. Diseño de la aplicación, se muestra como se pensó la aplicación, modulo a modulo, dando una explicación de cada uno y el funcionamiento de la misma. Detalles de la implementación, en este acápite detallamos los detalles interesantes de implementación, mostrando las herramientas auxiliares y como fue llevada a cabo la elaboración de la aplicación. Luego concluimos el documento y damos a conocer las posibles

recomendaciones de aquellos aspectos a desarrollar en el futuro para explotar al máximo este programa.

Estado del Arte:

Dada la gran cantidad de información de que se dispone tanto en el terreno de las ciencias, el comercio o la vida social es que se hace evidente la necesidad de contar con métodos que permitan procesar datos y extraer de los mismos información útil.

Para visualizar la información contenida en documentos de texto mediante gráficas existen diversas etapas que son de vital importancia: procesar la información, extraer los datos de interés y crear gráficos. En cada etapa existen diversas ramas de la investigación dedicadas a su estudio. En el caso del presente documento, para el desarrollo de la aplicación objeto de esta tesis, se realizó un estudio de estas ramas.

Procesar y extraer información:

La información puede estar contenida en diversos tipos de archivos: bases de datos, documentos de textos, hojas de cálculo, etc.; estos archivos de acuerdo a como almacenan los datos los podemos clasificar en dos tipos: estructurados o no estructurados. Archivos estructurados son datos almacenados con una estructura interna, fácil de localizar, donde se conoce que hay en cierta posición del archivo, lo podemos ver como un archivador perfectamente organizado estando todo identificado y etiquetado, como ejemplo de esto están algunas bases de datos y archivos de textos como *CSV*[1]*.* Mientras que los datos no estructurados son datos que no contienen estructura interna identificable, es un conjunto desorganizado de varios objetos que se almacenan sin valor alguno hasta que se identifican, ejemplo de ellos puede ser archivos de texto, hojas de cálculo, correos electrónicos, etc.

Cuando se parte de un documento del cual se desconoce su estructura y contenido en sí, se hace complejo poder extraer datos que le sean de utilidad al usuario. En este punto es donde se busca reconocer estructuras en los datos, acomodarlos para partir de algo conocido y extraer información que resulte valiosa. Actualmente existen ramas de la inteligencia artificial y de recuperación de información que estudian como extraer datos y procesarlos, transformándolos a estructuras comprensibles posteriormente, como es el caso de la minería de datos y minería de textos.

La minería de datos es el proceso de obtener conocimiento nuevo a partir de grandes cantidades de datos [3]. Es un campo interdisciplinario, al que contribuyen muchas áreas, como son: la estadística, el aprendizaje de máquinas, recuperación de la información, reconocimiento de patrones y bioinformática. El objetivo de la minería de datos es el

[1]Archivos de datos estructurados donde los valores son separados por comas (*Comma-separated values*).

descubrimiento de patrones, perfiles, anomalías y tendencias a través del análisis de los datos. Las principales técnicas en este campo son enfocadas al aprendizaje de máquinas.

Entre las áreas que abarca la minería de datos se encuentra la minería de textos, en la que los datos son documentos de texto. La minería de textos se caracteriza por poseer la información muy poco estructurada y confusa, mientras que la minería de datos trabaja con bases de datos estructuradas. Dentro de la minería de datos, la minería de textos es considerada como una de las áreas de mayor potencial.

Podemos definir la minería de textos como el proceso de extracción de patrones y conocimientos de interés y no triviales a partir de textos no estructurados. Es, por tanto, un campo multidisciplinario, que incluye conocimientos de recuperación de la información, análisis de textos, extracción de la información, *clustering*, categorización, aprendizaje de máquinas, etc. [4].

En este trabajo de tesis resulta de interés ver los pasos en los que se descomponen estas dos ramas. La minería de datos en general descompone en varias fases incluso se profundiza más en la extracción de conocimiento en sí. Esto específicamente no es objeto de nuestro trabajo no obstante sería interesante en un futuro profundizar en estas técnicas y extraer mayor cantidad de información con deducción de conocimiento. En la ejecución de nuestro programa nos hemos apoyado solo de las fases que a continuación nombramos:

1. Limpiar y pre-procesar los datos, acomodándolos.
2. Clasificar los datos según sus estructuras.
3. Normalizar los datos creando una estructura intermedia que sepamos trabajar.
4. Seleccionar datos interesantes.

Tanto la minería de textos, la minería de datos y la visualización de datos nos ayudan a comprender el proceso para reconocer estructuras y extraer los datos que creamos importantes para mostrarlos luego. Todo esto en su conjunto nos da una base para definir una arquitectura sustentada en módulos, guiándonos por las mismas etapas descritas, los cuales finalmente nos permitan reconocer estructuras en textos, extraer información descriptiva y mostrarla de la mejor forma posible.

(hablar de Parsers) Para reconocer estructuras en el texto es necesario estudiar el uso de gramáticas y expresiones regulares, con el fin de verificar la estructura de la información antes de pasar a descomponerla.

Bibliotecas del lenguaje utilizadas

De manera general vemos que hay ramas de la ciencia que se dedican al estudio de reconocimiento de estructuras, extracción de datos, información e incluso llegar a extraer

conocimiento en sí. Sin llegar a tanta profundidad por los objetivos del presente trabajo, nos interesa ver los pasos planteados de los que se compone la minería de datos, ver que tan importante es acomodar los datos para luego ver herramientas como gramáticas, expresiones regulares que nos permitan extraer datos, normalizándolos, para pasar a visualizarlos.

Graficar Datos:

A la hora de graficar un paso importante es definir la mejor manera de mostrar la información. Los gráficos ayudan mucho a comprender los datos numéricos de un texto. La rama del periodismo de datos se centra en estudiar como mostrar la información y como contar una historia con datos.

El periodismo de datos utiliza las herramientas estadísticas y de visualización de datos con el objetivo de contar las viejas historias de otra forma, de un modo más claro para el público, y descubrir otras evidencias nuevas, ocultas hasta entonces [5].

En la actualidad el periodismo de datos ha alcanzado gran éxito a nivel internacional. Para ello, partimos del hecho que las tecnologías de la información han cambiado por completo el concepto que teníamos del mundo. Se han creado miles de herramientas nuevas que han diversificado los usos de las que ya conocíamos. En esta vorágine de novedades, se comenzó a “mezclar” distintas herramientas para nuevos usos. La ciencia informática y la telemática han dado pie al surgimiento de un entorno profesional que se explota cada vez más en los medios [6].

Existen diversas herramientas tecnológicas disponibles que limpian, analizan y cruzan los datos de forma tal que el público pueda visualizarlos y entenderlos con facilidad. Algunos de los programas más utilizados por los periodistas de datos son: Data Wrangler, una aplicación de la Universidad de Standford que permite explotar hojas de cálculo; Google Fusion Tables, que convierte bases de datos en mapas y geolocaliza direcciones en ellos; Google Refine, que gracias a sus filtros depura la información, homogeneizando las bases de datos utilizadas; Statwing, recomendado para el análisis de datos estadísticos y cruces de variables; Data Wrapper, especialmente indicado para la creación de gráficos, y Piktochart, que también permite crear infografías [7].

Para Giannina Segnini, directora de la Maestría de las Ciencias de Periodismo de Datos de la Universidad de Columbia, Nueva York, en el periodismo de datos deben contemplarse cinco pasos básicos [8]:

1. Obtención de los datos.
2. Limpieza de los datos.
3. Análisis.

4. Verificación de la información.
5. Visualización.

Estos pasos son fundamentales a la hora de crear un gráfico, podemos asumir que la obtención de datos y limpieza de estos se llevan a cabo con el desarrollo del *data mining, fusionándose así* los primeros dos pasos. Con el paso 3 se eligen de todos los datos obtenidos los que creamos importantes, haciéndole análisis estadísticos, acorde al mensaje que se quiera transmitir. Luego un periodista verifica la veracidad de la información, y se pasa a la visualización.

Para visualizar los datos debemos elegir entre un conjunto de gráficos, esto puede ser engorroso pues una mala representación trae consigo una mala interpretación de los datos. Para ayudar a esta elección se categorizan los gráficos, acorde al mensaje que se quiera mostrar para luego seleccionar el tipo de gráfico que se adapte mejor a tus datos. Existen cuatro tipos de mensaje que podemos transmitir con datos: una relación, una distribución, una comparación y una composición, cada uno con un significado diferente:

1. Una relación muestra una conexión o una correlación entre una o varias variables que hemos analizado. Por ejemplo, si colocamos la edad de los trabajadores en el eje x y el número de hijos en el eje y, podemos ver si existe una conexión entre ellos.
2. Una distribución muestra una colección de datos (relacionados o no) para comprobar si existe alguna interacción entre ellos o si están correlacionados. Por ejemplo, el número de bajas maternales de cada mes durante el año.
3. Una comparación muestra cómo una serie de variables reaccionan frente a una variable común. Por ejemplo, el número de mujeres versus hombres en una empresa a lo largo de los años o el radio de rotación entre diferentes empresas.
4. Por último, una composición recoge diferentes tipos de variables que pueden englobarse dentro de un mismo umbral y las muestra todas juntas. Por ejemplo, si hemos realizado un test entre nuestros trabajadores y queremos mostrar el porcentaje de respuesta de cada una de las posibles respuestas.

[9], [10].

Teniendo seleccionado el grafico a mostrar, solo nos queda ver que herramienta nos permite mostrarla. Actualmente existen muchas herramientas para crear gráficos, y las mismas no son necesariamente enfocadas para el desarrollo de software como: Highcharts, D3, GoogleCharts.

De manera general, hasta aquí hemos podido ver los objetivos del periodismo de datos, conocer en qué pasos nos podemos apoyar a la hora de contar algo mediante gráficos; las diferentes formas de clasificar los gráficos, acorde el mensaje que se quiere transmitir para

luego seleccionar el grafico más adaptable a los datos. Además de ver algunas herramientas que se utilizan para visualizar los mismos.

Tecnocas para la selección

Herramientas de visualización y procesamiento de información:

Las herramientas para graficar datos son bastantes utilizadas y exigidas hoy en día. Existen diversas aplicaciones que permiten visualizar datos mediante gráficas, mapas, tablas, etc. Estas herramientas tienen fines diversos, algunas son simplemente librerías de lenguajes de programación, herramientas online, en su mayoría estos procesan la información de una base de datos o un archivo que conocen la estructura, dígase archivo CSV o archivos Excel. Entre los ejemplos más utilizados de aplicaciones tenemos: Tableau, un software que se especializa en técnicas de visualización, enfocado en inteligencia empresarial, analiza y procesa volúmenes de datos, se enfoca en datos bien estructurados, establece un formato, bases de datos relacionales y hojas de cálculo [11]; SPSS desarrollado por IBM el que ofrece un avanzado análisis estadístico, extensible con *open-source,* varios algoritmos de *matching-learning,* análisis de textos [12] muy usado en las ciencias sociales y exactas, este puede ser usado con múltiples tipos de bases de datos, bases de datos relacionales y textos, en caso de recibir texto el usuario especifica de qué forma se analiza, especificando la estructura [13]; LightningChart, herramienta de visualización de datos de Visual Studio[2]. La mayoría de estas herramientas no son exentas de pago, incluso muchas están enfocadas a algún campo, sean negocios u otro.

Herramientas más simples permiten generar gráficos de igual manera sin hacer minería de datos como **Infigr.am** que es una herramienta para crear infografías, reportes, gráficos, etc., donde se puede importar los datos de archivos CSV o XLS; **Google Charts,** una aplicación de Google para realizar estadística web con buen monto de gráficos; **Microsoft Excel** herramienta bien conocida que permite crear gráficos, pero como las anteriores, no hace pre-procesamiento de los datos, espera los datos en formatos establecidos y solo trabaja con datos estructurados o datos que se insertan de forma manual.

En resumen, podemos apreciar varias herramientas y métodos que se usan en la actualidad para cada parte de este proceso. Vemos que para procesar y extraer información de documentos la minería de datos en general está abordando estos aspectos, apoyándose principalmente en técnicas de inteligencia artificial. De igual forma para graficar el periodismo de datos utiliza varias herramientas y ha creado la necesidad del surgimiento de nuevas aplicaciones que facilitan la creación de gráficos e infografías fusionando varias de estas para

[2] Entorno de desarrollo integrado (IDLE por sus siglas en ingles) de Microsoft.

mostrar información y los pasos que deben contemplarse para mostrar buenos resultados en gráficos.

Diseño de la aplicación:

La aplicación consta de cinco módulos principales: el módulo de interacción con el usuario, el identificador de datos, una capa intermedia de formatos conocidos (KF), el módulo que genera los gráficos y un módulo extra para retroalimentar el proceso. En la Figura 1 se puede apreciar el flujo general de la aplicación:

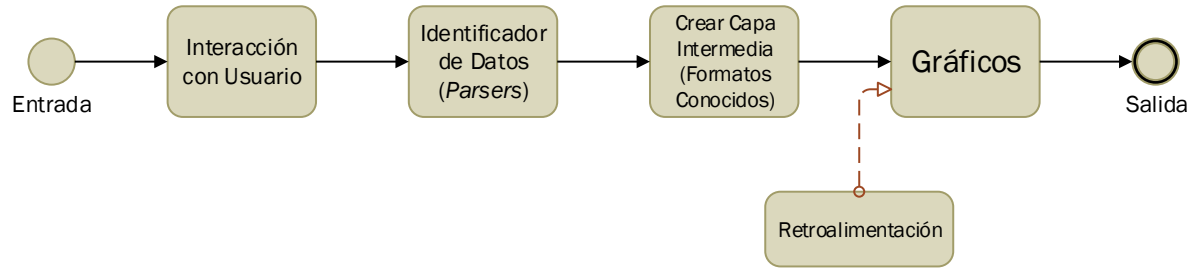

Figura 1. Diagrama de flujo general de la aplicación. Ver flecha de doble sentido en la retroalimentacion

A continuación, se detallan cada uno de los módulos de la aplicación.

Módulo de Interacción con el usuario:

El módulo de interacción con el usuario se encarga de procesar los parámetros de entrada del programa y establecer la configuración que tendrá el mismo durante todo el proceso.

Este módulo recibe los parámetros del usuario, y establece la configuración del programa, la cual se mantiene hasta el fin del proceso. Una vez establecida la configuración se pasa a mostrar ayudas, generar juegos de datos, o cargar ficheros para su análisis. Solo se realiza una opción en caso de solicitar varias, las mismas se realizarán de acuerdo al orden descrito. En caso de mostrar ayuda, el proceso termina en este módulo luego de mostrar la misma. Mientras que si se acciona generar juegos de datos se pasa a graficar o a generar datos en los módulos correspondientes (Gráficos o Identificador de Datos), dependiendo de la configuración y al generarlos, termina el proceso. En caso que se cargue el fichero, se pasa al Identificador de Datos continuando el flujo del programa hasta su fin. En la Figura 2 se puede apreciar un esquema del funcionamiento de este módulo.

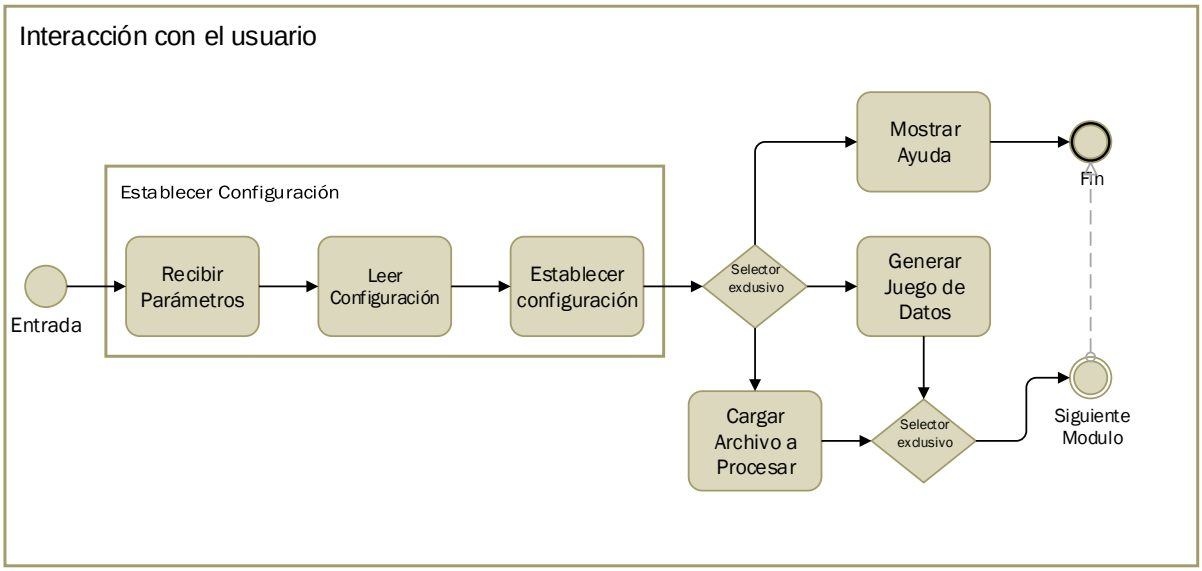

Figura 2. Esquema del flujo de la Interacción con el Usuario.

Módulo Identificador de Datos:

El identificador de datos trata de reconocer estructuras en la cadena creando formatos para graficarlos posteriormente.

El módulo recibe los datos como cadena de texto, donde primero hace un pre-procesamiento de estos acomodándolos para extraer información. Luego se pasa a analizar la cadena completa. De ser reconocida se construyen los formatos conocidos (KF) pertinentes. En caso contrario, se hace un procesamiento más profundo, al analizar línea a línea la cadena, y creamos los KF apropiados. Al analizar todas las líneas, y tener un conjunto de KF pasamos a compactarlos, tratando de unir los KF similares en líneas adyacentes (Figura 3).

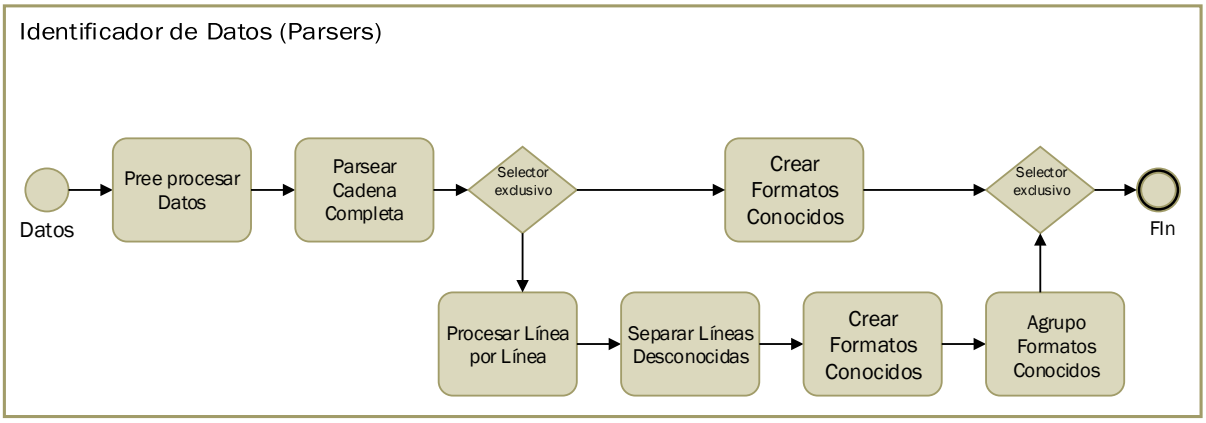

Figura 3. Esquema del flujo del Identificador de datos.

Módulo Formatos Conocidos (KF):

Los Formatos Conocidos constituyen una capa intermedia. Este módulo está compuesto solamente por objetos que definimos como formatos conocidos, independientes de otro módulo en sí, pero a la vez constituyen los pilares de la aplicación. Todos los *parsers* saben que tipos “KF” generan, y cada grafico sabe que tipos de “KF” aceptan para poder graficar (Figura 4).

Figura 4. Esquema del Módulo Formatos conocidos.

Los Formatos conocidos son estructuras que sirven como capa traductora de un módulo al otro, pues se necesita transformar la información extraída a un objeto que se sepa como graficar. De esta forma se definen un conjunto de formatos conocidos de acuerdo a los tipos de datos diferentes que se necesiten para poder graficar información, mientras que los *parsers* son capaces de crear varios formatos conocidos acorde a la información que procesan. La Figura 5 muestra la relación entre un *parser* y un formato conocido. Mientras que la Figura 6 evidencia cómo un formato conocido puede generar uno o más gráficos.

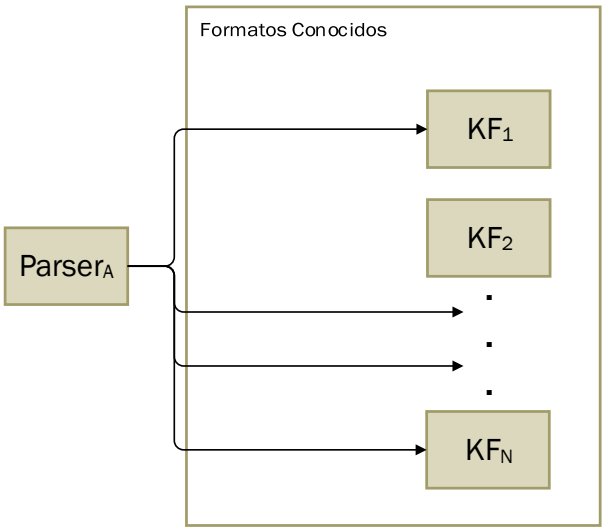

Figura 5. Relación entre un Parser$_A$ y los Formatos conocidos.

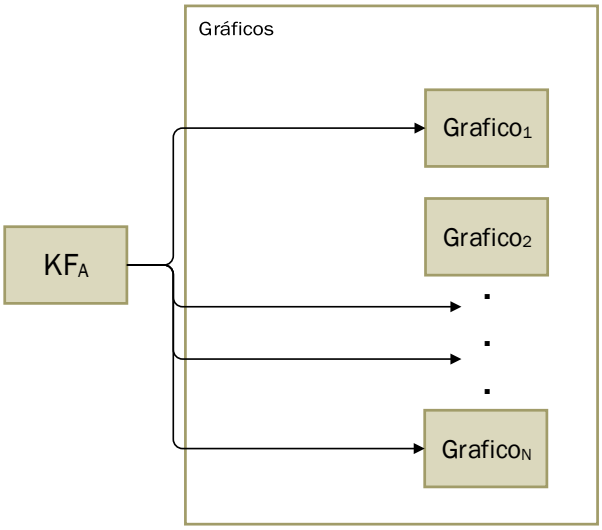

Figura 6. Relación entre un Formato Conocido (KF_A) y los Gráficos.

Módulo Graficar (Generación de gráficos):

Este módulo está compuesto por un conjunto de gráficos definidos. Estos parten de un formato conocido y el objetivo del módulo es crear todos los gráficos posibles como se puede apreciar en la Figura 7:

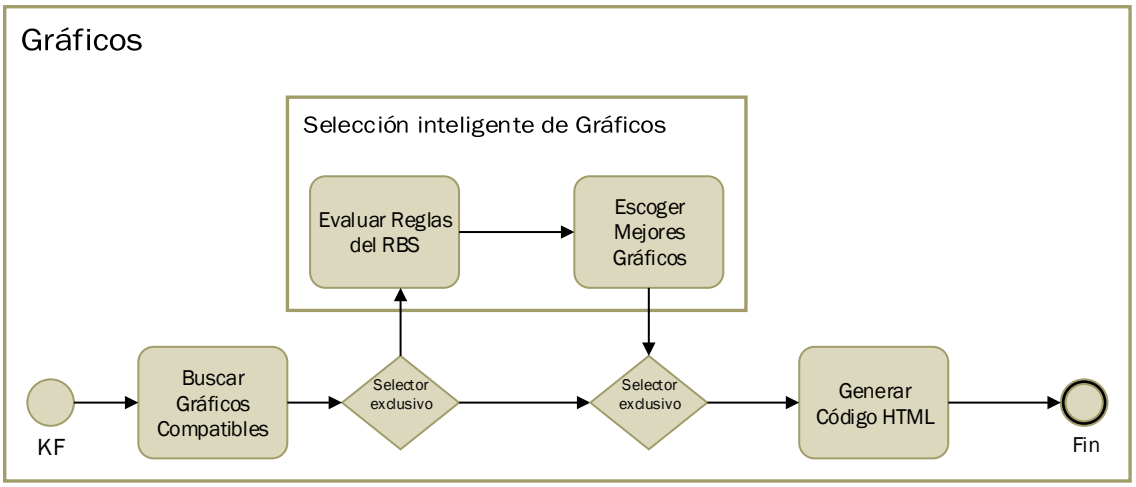

Figura 7. Esquema del flujo de Graficar.

Primero se buscan todos los gráficos compatibles con el formato conocido, luego en dependencia de la configuración del programa, se seleccionan los gráficos que más se asemejan a los datos o se prosigue con el conjunto de gráficos inicial. A continuación, se genera el código de los gráficos para luego construir el archivo de salida y concluir el programa.

La selección de Inteligente de Gráficos está diseñada con ideas de un Sistema basado en reglas (SBR). En la Figura 8 se explica su diseño. Este sub-módulo primero evalúa las reglas en los gráficos seleccionados, contando con la retroalimentación o *feedback* de los gráficos que han sido útiles para el usuario. De igual forma se evalúan las reglas para ver qué tipo de mensaje se quiere transmitir contando con el *feedback* de estos, así se selecciona el mensaje a transmitir y se termina de escoger los k-mejores gráficos para mostrar.

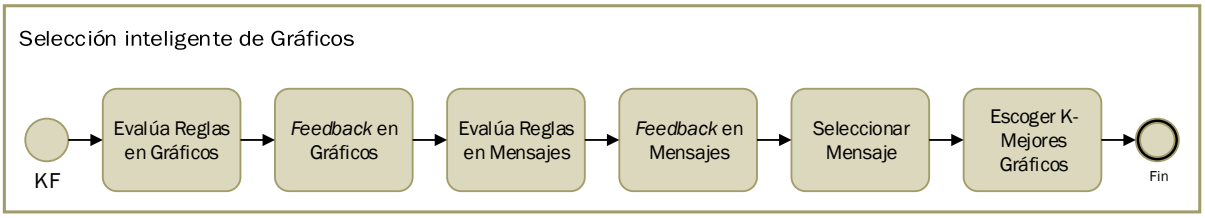

Figura 8. Descripción del paso llamado Evaluar Reglas del Módulo Graficar.

Retroalimentación:

Este módulo es una pequeña aplicación que muestra un archivo que se ha generado en el proceso principal y permite al usuario seleccionar los gráficos que fueron de utilidad guardando esta información en una base de datos, para utilizarla en próximas ejecuciones para seleccionar con más precisión los gráficos a graficar.

De esta forma vemos como el fuljo principal del programa se resume en recibir parámetros de la interacción con el usuario y establecer la configuración, cargar el fichero a procesar, proceder a limpiarlo y acomodarlo para su análisis, examinar el texto completo y crear estructuras conocidas, para en un final seleccionar los mejores gráficos y mostrarlos al usuario. Finalmente crear el fichero de salida; teniendo una pequeña aplicación que nos permite abrir el archivo y seleccionar si fue útil o no.

Detalles de Implementación:

La aplicación es implementada en el lenguaje de programación *Python.*

Ya viendo el flujo de la aplicación veremos los detalles de implementación de la misma. Para esto partimos de la estructura y damos un recorrido por los principales módulos de la aplicación. Se irán explicando cómo están distribuidos, implementados, las herramientas de las que se auxilian y sus aspectos esenciales que garantizan el funcionamiento del software.

Estructura

Para explicar el flujo del programa (Figura 1), mostraremos cómo fue implementado, examinando su estructura y distribución.

La raíz de la aplicación consta con un archivo “an.py”, que es el archivo que se debe correr para iniciar el flujo del programa, un archivo “input_parser.py” y un archivo “config.ini” que se explicaran más adelante. La carpeta “api” contiene los módulos principales del programa, “graphs”, “known_formats” y “parsers”, estos son gestionados por “an_graphs.py”, “an_identify.py”, “an_known_format.py”, respectivamente. Contiene una carpeta “utils” la cual posee archivos útiles para la implementación de la app y una base de datos de la cual nos apoyamos más adelante para poder retroalimentar la aplicación. Las carpetas “data_generator” y “out” contienen los archivos con juegos de datos que se generan y los archivos resultantes al terminar el proceso. Por ultimo tenemos la carpeta “load_html” que es la encargada de la retroalimentación. La Figura 9 muestra la distribución de la aplicación.

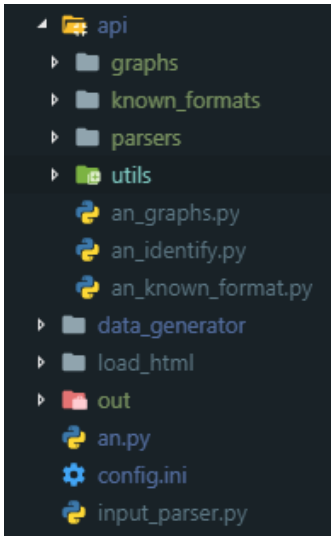

Figura 9. Distribución de la aplicación.

Interacción con el usuario (CLI):

La interacción con el usuario es la encargada de leer los parámetros de entrada y establecer la configuración de la aplicación, es la primera etapa del programa.

El archivo “input_parser.py” es el encargado del módulo de interacción con el usuario, el que esta implementado con una interfaz de líneas de comando (CLI por sus siglas en inglés). Se apoya de la librería *argparser*[3] y así nuestra interfaz recibe un monto de parámetros y ofrece una ayuda al usuario. Un ejemplo de interacción con esta interfaz se ve a continuación en la Figura 10.

```
PS D:\Documentos\GitHub\Analyser> python .\an.py --parser p_numbers_list
====================
```

Figura 10. Interacción con el CLI, recibiendo parámetro “parser” con valor p_numbers_list.

Acorde a los parámetros recibidos la aplicación establece la configuración y pasa a mostrar ayuda, generar juegos de datos o analizar un archivo, acorde al pedido del usuario. Para establecer la configuración, nos apoyamos en una clase “Config()” la que inicialmente contiene los parámetros por defecto de la aplicación y almacenará la configuración de la app durante todo el flujo de la misma. Para establecer los parámetros de esta clase primero se busca la existencia de un fichero “ *config.ini* ”, en caso de existir, leemos sus parámetros y actualizamos la clase, luego tomamos los parámetros de la entrada estándar y sobrescribimos los anteriores, estableciendo claramente una prioridad en el proceso de ajuste de la configuración: primera prioridad son los parámetros de la entrada estándar, seguido del archivo config.ini, y en un final los parámetros definidos por defecto en “*Config()*”.

El archivo “config.ini”, es un archivo compuesto por secciones (establecidas entre corchetes) y valores (los cuales se definen parámetros y su valor), lo cual facilita la extracción de sus valores. Se ve un ejemplo en la Figura 11. Este archivo no tiene necesidad de existir, en ese caso se establece la configuración acorde los parámetros de la entrada estándar. No todos los valores tienen que estar presentes en el archivo.

```
[PARSERS]
path = ./api/parsers
available = p_csv
help =
list= 0
```

Figura 11. Ejemplo de archivo “config.ini” donde se establece la sección PARSERS y tiene parámetros “path”, “available”, “help”, “list”, con sus respectivos valores.

[3] Es un módulo de *parseo* por comando de línea recomendado por la librería estándar de Python. [14]

El flujo de configuración se puede ejemplificar si se asume que la entrada es la mostrada en Figura 10 y el archivo "*config.ini*" es el mostrado en Figura 11, por la entrada estándar el usuario nos pide que se analicen los datos solamente con el *parser p_numbers_list,* mientras que en el fichero está habilitado el *parser* p_csv, por lo antes expuesto la configuración final tendrá como *parser* habilitado *p_numbers_list.*

La clase "*Config()"* tiene como singularidad que implementa el patrón *singleton*, así solo se puede crear una instancia de la misma y proporciona un acceso global a esta.

Identificador de Datos:

El Identificador de Datos es el encargado de reconocer estructuras en la cadena a procesar y extraer la información que luego se graficará. Se analizará su composición, las herramientas utilizadas y una explicación del flujo del módulo, expuesto en la Figura 3, haciendo hincapié en la implementación del mismo.

Composición

El módulo está compuesto por un conjunto de *parsers,* métodos para limpiar la cadena y un conjunto de métodos para procesar la información y extraer los datos que nos resulten de interés. Esta fase del programa es administrada por el archivo "*an_identify.py".*

Los *parsers* son clases del lenguaje que tienen como objetivo reconocer estructuras, extraer datos y normalizarlos creando Formatos Conocidos. Deben cumplir ciertos requisitos para poder ser utilizados en la aplicación: el nombre del archivo debe seguir el siguiente formato: *"p_nombre_clase_parser.py"*, de igual forma la clase se debe llamar *"NombreClaseParser"*, se estandarizaron estos aspectos para facilitar el desarrollo de la aplicación. De igual forma las clases deben implementar la interfaz que se puede ver en la Figura 12, de no cumplir con estos requisitos, el archivo no se considerara un *parser* válido y no se tendrá en cuenta en el curso de la aplicación. De forma opcional, un parser pude tener un método para generar

juegos de datos, de existir el método se debe nombrar *"data_generator"* con la estructura que se muestra en la Figura 13:

```
class IParser:
    def parse(self,text_to_process:str):
        ''' Recibe el texto a procesar  y retorna una lista de Formatos Conocidos
        acorde a los datos que extrajo'''
        pass
    def help(self):
        ''' Muestra la ayuda del Parser con un ejemplo del texto que reconoce
        return -> str '''
        pass
```

Figura 12. Interfaz que debe implementar un parser definido.

```
def data_generator(self, path, amount=50, on_top=50, below=100):
    ''' Genera juego de datos con el formato que reconoce el parser
    amount= 50 cantidad de series
    on_top=50, below=100 para generar números X on_top <= X <= below
    Donde crea un archivo con los datos generados en la ruta descrita en la configuración
    '''
    pass
```

Figura 13. Método para generar juegos de datos.

Siguiendo las buenas prácticas de programación, este módulo se desarrolló de forma tal que su extensibilidad fuera bastante sencilla. En el módulo se importan dinámicamente los *parsers* definidos en el directorio que se estableció en la configuración, por defecto el directorio establecido es ". */api/parsers"*, así para adicionar nuevos *parsers* solo se debe crear el nuevo fichero *".py",* con el formato explicado anteriormente y de esta forma ser utilizado por el programa.

Nuevos métodos de procesar el archivo también pueden ser agregados con facilidad, luego de tener su implementación, se procedería a agregarlos al archivo principal del módulo y adicionarlo a la lista "*parsers_phases*" que se recorre para analizar los datos con cada método.

Herramientas auxiliares

Para analizar las cadenas y verificar las estructuras de las mismas los *parsers* definidos se auxilian de gramáticas o expresiones regulares para reconocer estructuras, para esto se utilizaron las librerías *"ply"* y la librería *"re"*, respectivamente. En el uso de gramaticas, ply trabaja con gramaticas libres del contexto de la forma BFN.

Flujo

Auxiliándonos de la Figura 3 para la explicación de esta fase del programa, se describe a continuación detalladamente como está concebido el flujo del mismo.

Para el funcionamiento de esta etapa se importan dinámicamente los *parsers* definidos en la ruta establecida en la configuración. Luego se pasa a limpiar la cadena donde se eliminan las líneas en blanco, y los caracteres especiales (*áéíóúüñ/;()!@#$%^&*?><~`*). Con la cadena acomodada se recorren todos los métodos de análisis y extracción de información definidos (actualmente son dos, se tiene la idea futura de incrementar este número) y se procesa la cadena creando formatos conocidos.

Para procesar el texto tenemos dos métodos, uno que toma la cadena de texto completa para ver si algún *parser* habilitado la reconoce y normaliza la información y otro más profundo que va analizando línea a línea del texto inicial, y procesa esa línea con todos los *parsers* habilitados, y va creando KF por cada línea, al terminar con el texto completo recorre todos los KF creados y si en líneas adyacentes hay formatos conocidos similares los compacta.

De esta forma obtenemos un módulo fácilmente extensible, el que trata de reconocer estructuras dentro de un texto el cual no sabemos cómo está conformado. Extrae los datos que reconoce creando formatos conocidos.

Formatos Conocidos (KF)

Los Formatos Conocidos son las estructuras de datos que, definidas para normalizar los datos, creándose de información textual por la fase anterior para poder analizarlos mejor y poder graficarlos luego, conforman la capa intermedia de la aplicación, teniendo gran importancia en el funcionamiento de la misma. En este acápite se mostrará como fueron implementados estas estructuras, y que modelo se debe seguir para la implementación de nuevas estructuras.

Los Formatos Conocidos están implementados en la ruta *"./api/known_formats/"*. Estos son clases que heredan de *"KnownF()"* la que podemos ver en la Figura 14. Estos están compuestos por la propiedad *"elements"* que almacena en un diccionario las series de os futuros graficos con sus respectivos nombres, otras propiedades la cantidad de series que almacena. Tienen un método *"extend"* que su objetivo es poder extender dos formatos conocidos iguales, agregando los elementos de uno en otro y actualizando las propiedades.

```python
class KnownF:
    def __init__(self, s_values=[],series_names=[]):
        self.elements = {}  #almacena las series, las llaves son los nombres de estas
        self.min_value = 9999999999 #valor mínimo
        self.max_value = -999999999    #valor máximo
        self.count = len(s_values)  #cantidad de series
        self.categories = []    #nombre de las categorías
        self.keys = []
    def extend(self,kf_to_extend):
        ''' Extiende el KF self, con el KF kf_to_extend
        agregando a los elements los de kf_to_extend
        si alguna serie tiene el mimso nombre se le cambia el nombre a esta para no sobreescribir
        actualiza el minimo, maximo, cantidad de series '''
```

Figura 14. Clase KnownF, muestra la estructura necesaria de los Formatos Conocidos, la clase padre de los KF ya implementados.

Graficar (Generación de gráficos):

Graficar es la etapa final del programa, en esta se parte de un formato conocido y la intención es generar las gráficas necesarias para mostrar la información y seleccionar cuales son las mejores graficas acorde a los datos extraídos. En esta sección se explicara la composición de esta fase, las herramientas auxiliares, el modelo presentado de selección inteligente de gráficos, el flujo expuesto en la Figura 7 haciendo hincapié en la implementación del mismo.

Composición

El módulo está compuesto por un conjunto de gráficos y un sub-módulo de selección inteligente de gráficos. Esta fase del programa es administrada por el archivo "*an_graphs.py".*

Los gráficos son clases del lenguaje que hemos definido, el objetivo de estas clases es apoyarse en los KF para crear el grafico pertinente a la clase, actualmente solo genera código para crear gráficos en HTML, pero también se pudiera añadir otros tipos de salida.

Cada gráfico debe cumplir ciertos requisitos para poder ser utilizados en la aplicación: el nombre del archivo debe seguir el siguiente formato: *"g_nombre_clase_grafico.py"*, de igual forma la clase se debe llamar *"NombreClaseGrafico"*. Estos aspectos se estandarizaron para facilitar el desarrollo de la aplicación. Por otro lado las clases deben implementar la interfaz que se puede ver en la Figura 15, si no cumple con estos requisitos el archivo no se considerara un gráfico válido y no se tendrá en cuenta en el cursar de la aplicación. De forma opcional un gráfico pude tener un método para generar un gráfico con valores aleatorios, de existir el método se debe nombrar *"generate"* con la estructura que se muestra en la Figura 16.

```python
class IGraph:
    def __init__(self):
        self.type='Line'
        self.message=[]
        self.kf_permited =[]

    def graphic(self,g_id, format_known):
        ''' Recibe el id que tendra el grafico, y los datos a graficar KF
        returna el codigo JS para generar los graficos en un HTML'''
        pass

   def evaluate_rules(self, path, amount=50, on_top=50, below=100):
        ''' Se definen las reglas para las cuales el gráfico obtiene punto
       en la selección inteligente de gráficos
        '''
        pass
```

Figura 15. Interfaz que deben implementar los gráficos "type" es el tipo de gráfico, "message" es una lista con los tipos de mensaje que transmite el grafico, "kf_permited" una lista con los Formatos Conocidos que puede procesar.

```python
def generate(self, id):
    '''Genera un gráfico con valores aleatorios '''
    pass
```

Figura 16. Método para generar gráficos con valores aleatorios. Donde recibe id como parámetro que es el id del grafico a generar.

Añadir nuevos gráficos a la estructura es bastante sencillo, solo se tiene que crear el archivo y colocarlo en el directorio del cual se están leyendo los gráficos. Este módulo importa dinámicamente los gráficos definidos en el directorio que se estableció en la configuración. Por defecto el directorio establecido es ". */api/graphs"*.

Herramientas auxiliares

Para la implementación de este módulo nos apoyamos de la biblioteca de java script highcharts, una librería con un amplio conjunto de gráficos y mapas, fácil de implementar. En caso de añadir nuevos gráficos e infografías no es un requisito auxiliarse de esta.

Selección Inteligente de Gráficos

A la hora de graficar tenemos muchos gráficos donde cada uno transmite con los mismos datos puntos de vista diferentes, mensajes diferentes, en este sub-módulo se plantea una estrategia para seleccionar la mejor propuesta de grafico para visualizar los datos, basándonos en un Sistema Basado en Reglas (SBR). Para ello clasificamos los gráficos, acorde al mensaje que se quiera transmitir. Donde un gráfico puede transmitir más de un

mensaje. Existen cuatro tipos de mensaje que podemos transmitir con datos: una relación, una distribución, una comparación y una composición.

Normalmente cuando se quiere crear un gráfico, el humano sabe el mensaje que quiere transmitir, acomoda los datos y selecciona el grafico que mejor le transmita el mensaje,

La idea tras el SBR es implementar un conjunto de reglas (métodos) que dado un Formato Conocido nos calcule los gráficos más cercanos a nuestros datos

Flujo

Auxiliándonos de la Figura 7 para la explicación de esta fase del programa, se describe a continuación detalladamente como está concebido el flujo del mismo.

Para el funcionamiento de esta etapa se importan dinámicamente los gráficos definidos en la ruta establecida en la configuración. Asumiendo que la selección inteligente esta activada, pasamos a ver auxiliándonos en el SBR cuáles son los gráficos que mejor visualizan nuestros datos para generarlos, en caso que no esté activada esta selección, se buscaran todos los gráficos que aceptan este KF y se genera su gráfica.

De esta forma obtenemos un módulo fácilmente extensible, el que trata de generar los mejores gráficos, donde se pueden agregar tanto gráficos como reglas a nuestro SBR fácilmente para mejoras futuras.

Retroalimentación

Requerimientos

python 3, configparser, json, electron, re, ply, highcharts

Recomendaciones

Conclusiones:

Bibliografía

[1] J. M. Chambers, W. S. Cleveland , B. Kleiner y P. A. Tukey, Graphical Methods for Data Analysis, Florida: CRC Press, 2018.

[2] D. Kelly, J. Jasperse y I. Westbrooke, «Designing science graphs for data analysis and presentation. The bad, the good and the better,» Science & Technical Publishing, New Zealand Department of Conservation, 2005.

[3] J. Han, M. Kamber y J. Pei, Data Minins Concepts and Techniques, Watham: Elsevier Inc, 2012.

[4] E. A. Doctor Bracho, «Técnicas Estadísticas en Minería de Textos,» Universidad de Sevilla, Sevilla, 2018.

[5] J. Flores Vivar y C. Salinas Aguilar, «Sinergias en la construcción del Nuevo Periodismo derivadas del Data Journalism y el Transmedia Journalism,» Actas III Congreso Internacional Comunicación 3.0, Salamanca, 2012.

[6] J. Flores Vivar y C. Salinas Aguilar, «El periodismo de datos como especialización de las organizaciones de noticias en Internet,» *Correspondencias & Análisis,* pp. 15-36, 2013.

[7] M. Á. Chaparro Domínguez, «Nuevas formas informativas: el periodismo de datos y su enseñanza en el contexto universitario,» Universidad Internacional de La Rioja, 2014.

[8] F. Ramírez, «LAPATRIA.com,» 15 05 2019. [En línea]. Available: http://www.lapatria.com/blogs/periodismo-de-datos-periodismo-de-siempre.

[9] G. Zelazny, Say it with Charts, The McGraw-Hill Companies, Inc., 2001.

[10] M. Tatay, «Prisma,» 10 agosto 2016. [En línea]. Available: beprisma.com/como-elegir-la-mejor-grafica-mostrar-datos.

[11] Tableau, «Tableau,» 15 05 2019. [En línea]. Available: http://www.tableau.com/learn/get-started/data-structure.

[12] IBM, «IBM,» 15 05 2019. [En línea]. Available: http://ibm.com/products/spss-statstics.

[13] IBM, «Open source and IBM SPSS modeler The Best of the word,» IBM Analytics.

[14] G. Van Rossum, «Argparse Tutorial,» Python Software Foundation, 2018.