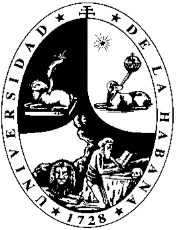
Facultad de Matemática y Computación

Universidad de La Habana.



Tesis de Diploma de la Especialidad Ciencias de la Computación

**Nombre de la tesis**

Autor: Elias Bestard Lorigados

e.bestard@estudiantes.matcom.uh.cu

Tutor: Pedro Quinteras Rojas

La Habana, 2019

Tabla de contenidos

1. Presentación
2. Agradecimientos y dedicatorias
3. Abstract (resumen)
4. Tabla de contenidos
5. Tabla de gráficos (opcional)
6. Tabla de tablas (opcional)
7. Introducción
   1. Motivación
   2. Antecedentes
   3. Objetivos
   4. Estructura del documento
8. Estado del arte (puede ir dentro de la introducción si es pequeño)
   1. Introducción
   2. Herramientas de extracción (reconocimiento) de datos
   3. Herramientas de graficacion
   4. Herramientas como esta
   5. Algoritmos, técnicas de aprendizaje, etc.
9. Marco teórico (opcional) (puede ir, diluido, en el capítulo de diseño si es corto)
   1. Definiciones
   2. Teoremas
   3. Contextos
   4. Definir una base, un lenguaje común para el lector y el escritor
10. Diseño de la aplicación, modelación, parte lógica
    1. Cli (esto creo que va en la implementación)
       1. Configuraciones
       2. Fichero
       3. Valores por defecto
    2. Parsing
    3. Capa intermedia de datos conocidos
    4. Graficar
    5. Módulo de aprendizaje
    6. Ensamblar todo eso
11. Detalles de implementación (implementación)
12. Pruebas (opcional)
13. Conclusiones
14. Recomendaciones
15. Bibliografía
16. Anexos

# Introducción

## Objetivo general

Desarrollar un *software* que permita graficar información a partir de documentos de los cuales se pueda desconocer cómo está estructurado.

## 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 aprendizaje sencillo que permita seleccionar de forma inteligente gráficos partiendo de un conjunto de datos.
4. Implementar un prototipo del modelo establecido con una interfaz de usuario amigable, (CLI).
5. Validar el prototipo presentado.

# Estado del Arte:

Para desarrollar

Para mostrar la información que contiene un archivo de texto en gráficas, es necesario procesar el documento y extraerle la mayor cantidad de información posible. Cuando se parte de un conjunto de datos, de los que no se tiene dominio de su estructura ni de su contenido en sí, llega a ser un poco complejo poder extraer datos que le sean de utilidad al usuario. En este punto donde se busca reconocer estructuras en los datos, acomodarlos para poder trabajar partiendo de algo conocido. Actualmente existen ramas de la inteligencia artificial y de recuperación de información que estudian como extraer datos y procesarlos, transformándolos a estructuras comprensibles posteriormente.

La minería de textos, minería de datos. Son ramas que trabajan sobre la extracción de datos, reconocimiento de patrones, entre otras, en documentos planos.

(Doctor Bracho, 2018)

La minería de datos es el proceso de obtener conocimiento nuevo a partir de grandes cantidades de datos (Han and Kamber, 2000). Es un campo interdisciplinar con contribuciones de 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, por tanto, es el descubrimiento de patrones, perfiles, anomalías y tendencias a través del análisis de los datos.

Las fases de la extracción de conocimiento a partir de bases de datos (KDD, Knowledge Discovery from Databases) son las siguientes (Han and Kamber, 2000):

1. Limpieza de datos, en la que se elimina el ruido y los datos inconsistentes obtenidos de las fuentes externas usando métodos estadísticos.
2. Integración de datos, en la que se combinan las distintas fuentes de datos.
3. Selección de datos, pues los datos almacenados en la base de datos pueden no ser todos de interés.
4. Transformación de los datos a un formato apropiado para la minera, pudiéndose optar por distintos tipos de modelos.
5. Minera de datos, en la que se aplican las técnicas pertinentes, en función de los datos y de la información que se desea extraer, para obtener patrones de datos.
6. Evaluación de los patrones obtenidos en el proceso de minera de datos, identificando los que resultan interesantes y los que no.
7. Representación del conocimiento.

Las principales técnicas en este campo incluyen clasificación y predicción, *clustering*, detección de *outliers*, reglas de asociación, análisis secuencial, análisis de series tempo reales, redes neuronales, arboles de decisión, etc. También son frecuentes técnicas más novedosas como el análisis de redes sociales o el análisis de sentimientos.

Dada la enorme variabilidad de los datos disponibles y de los cuales podemos extraer información de interés, la minera de datos incluye muchas áreas diferentes en función del tipo de datos a analizar. Entre estas áreas se encuentra la minera de textos, en la que los datos son documentos de texto, partiendo por esta razón de información muy poco estructurada.

La minería de textos es considerada como una de las áreas con mayor potencial dentro de la minería de datos, debido a que la manera habitual de almacenar información es en forma de textos. La minería de textos es, sin embargo, una disciplina mucho más compleja que la minería de datos, que trata con bases de datos estructurados. Esta tarea, por el contrario, requiere tratar con datos textuales que son propiamente desestructurados y más confusos.

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 multidisciplinar, que incluye conocimientos de recuperación de la información, análisis de textos, extracción de la información, clustering, categorización, aprendizaje de máquinas, etc.

(Doctor Bracho, 2018)

(ver otros aspectos de procesamiento, estructuración y extracción de información) mostrar programas que se usan para minería de textos(Wikipedia)

Graficar Datos:

Una vez que se tiene la información estructurada y procesada, se busca la mejor manera de mostrarla, tratando de evitar mostrar info sin valor, llegamos a un punto donde llega a ser un poco complejo de elegir la manera en la que se mostrara la información. Los gráficos ayudan mucho a comprender los datos numéricos de un texto, ya sean comparaciones de varias variables, mostrar la distribución de una variable o la relación de 2 de ellas o incluso como está compuesta cierta categoría. La rama del periodismo de datos se centra en estudiar como mostrar la información, como contar una historia con datos.

Los preoximo es del pdf (Nuevas formas informativas: el periodismo de datos y su enseñanza en el contexto universitario María Ángeles Chaparro Domínguez 45009-70969-4-pb.pdf)

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 nuevas, ocultas hasta entonces (Flores, 2012: 479). Es una combinación de varios factores, como son el olfato periodístico tradicional, la habilidad de contar historias y la ingente cantidad de información disponible en Internet. El periodismo de datos puede ayudar a los periodistas a contar historias complejas mediante visualizaciones de datos en forma de infografías (Gray, Bounegru y Chambers, 2012).

Para Mar Cabra, una de las periodistas españolas que desarrolla este tipo de información, “se trata de recabar y analizar grandes cantidades de información y datos detallados para después hacerlos comprensibles a la audiencia a través de artículos, visualizaciones o aplicaciones” (Servera, 2012).

Estas herramientas tecnológicas disponibles en línea, que pueden ser gratuitas o de pago, limpian, analizan y cruzan los datos. Además, permiten visualizarlos de forma que el público los entienda, independientemente de cuál su nivel formativo con respecto a las TIC. Alguno 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.

FIN (Nuevas formas informativas: el periodismo de datos y su enseñanza en el contexto universitario

María Ángeles Chaparro Domínguez 45009-70969-4-pb.pdf)

Lo Proximo es de (El periodismo de datos como especialización de las organizaciones de noticias en Internet, Jesús Flores Vivar, Cecilia Salinas Aguilar, Dialnet-ElPeriodismoDeDatosComoEspecializacionDeLasOrganiz-4739290.pdf)

Para Giannina Segnini (ápud Ramírez, 2012), el periodismo de datos no remplaza las formas tradicionales del oficio. Sólo es una herramienta para hacer mejor el trabajo por lo que plantea que en el periodismo de datos deben contemplarse cinco pasos básicos:

1. Obtención de los datos.
2. Limpieza de los datos.
3. Análisis.
4. Verificación de la información.
5. Visualización.

FIN(El periodismo de datos como especialización de las organizaciones de noticias en Internet, Jesús Flores Vivar, Cecilia Salinas Aguilar, Dialnet-ElPeriodismoDeDatosComoEspecializacionDeLasOrganiz-4739290.pdf)

Para generar gráficos existen muchos programas que se dedican a generar gráficos partiendo de un CSV o un archivo Excel. (Ejemplificar programas que muestran datos con breve explicación)

## Diseño de la aplicación:

La aplicación consta de cuatro módulos principales: el módulo de CLI, el identificador de estructuras, una capa intermedia de formatos conocidos (KF) y el modulo que genera los gráficos. En la Figura 1 se puede apreciar el flujo general de la aplicación:



Figura 1. Diagrama de flujo general de la aplicación.

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

## Módulo CLI:

El CLI es el encargado de procesar los parámetros de entrada del programa y establecer la configuración que tendrá el mismo durante todo el proceso. La Figura 2 describe el funcionamiento detallado del este módulo:



Fig. 2. Módulo CLI.

Este módulo es el encargado de establecer la configuración del programa, para esto se apoya de la clase “*Config()*” la cual 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()*”.

Ya con la configuración establecida hay tres opciones: mostrar ayudas, generar juegos de datos, o carga fichero y analizarlo. Solo se realiza una opción y en caso de solicitar varias, las mismas se realizarán de acuerdo al orden descrito.

### Modulo Identificador de Datos:

Este módulo está compuesto por una serie de parsers, un conjunto de métodos para analizar la información y otro conjunto de métodos para acomodar la cadena. El modulo trata de reconocer la cadena creando formatos para graficarlos posteriormente. La Figura 3 muestra detalladamente el flujo de este módulo.

El módulo recibe los datos como cadena de texto, donde primero hace un pre-procesamiento de la misma eliminando líneas en blanco, comentarios, caracteres especiales y tildes. Una vez terminada esta etapa se pasa a analizar la cadena completa, para esto se analiza si algún parser definido la reconoce. 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 examinar si algún parser reconoce la línea, en caso de reconocerla creamos los KF apropiados. Al analizar todas las líneas, y tener un conjunto de KF por cada línea pasamos a compactarlos, tratando de unir los KF similares en líneas adyacentes.



Fig. 3. Flujo del Módulo Identificador de datos. Donde X representa un XOR, KF: formato conocido

### Modulo Formatos Conocidos (KF):

Los KF constituyen una capa intermedia, es la base de la aplicación. Todos los parsers saben que “KF” generan, y cada Grafico sabe que KF acepta para poder generar el código “html” pertinente, de esta manera se define un conjunto de estos formatos (Figura 4).



Fig. 4. Modulo Formatos conocidos.

### Modulo Graficar (Generación de gráficos):

Este último módulo está compuesto por una serie de gráficos definidos. Estos parten de un formato conocido y su objetivo es crear todos los gráficos posibles como vemos a continuación en la Figura 5:



Fig. 5. Descripción del Modulo Graficar

Primero se buscan todos los gráficos compatibles con el formato, luego en dependencia de si el usuario habilito o no la selección de gráficos inteligentes, este genera el código de los gráficos para en un final crear el archivo htlml y terminar el programa.

La selección de gráficos inteligente esta diseñada con ideas de un Sistema basado en reglas. En la Figura 6 se explica el paso Evaluar Reglas del RBS que forma parte del Modulo Graficar.



Fig. 6. Descripción del paso llamado Evaluar Reglas del Modulo Graficar.

Como vemos en el flujo de este submódulo, primero se evalúan las reglas de los gráficos, donde estas nos dan una puntuación, a la cual se le agrega una relación de los gráficos que le han sido útiles al usuario que fueron realizados con anterioridad con ese mismo tipo de “KF” con respecto al total de gráficos de su tipo, esto es lo que se nombra como “feedback en gráficos”. Así pasamos a evaluar las reglas para elegir de igual forma que tipo de mensaje se trata de mostrar dado los datos analizados, que de igual manera cuenta con un paso de retroalimentación. A detectar qué tipo de mensaje se cree que se quiere transmitir se aumenta la puntuación a los gráficos que transmiten ese mensaje y de esta forma finalmente elegir los k-mejores gráficos acorde a la puntuación adquirida y poder generar su código.

# Explicación de la APP:

“Explicar la app detalladamente”

Para explicar el flujo del programa visto en la fig 1, vamos a estudiar cómo fue implementado este, viendo su estructura y distribución.

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. En 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 contiene 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 7 muestra la distribución de la aplicación.

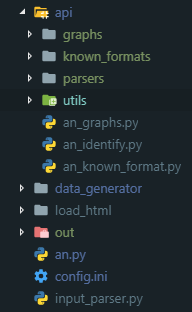


Fig 7 Distribución de la aplicación.

El archivo “input\_parser.py” es el encargado del módulo CLI, este se apoya de la librería argparser, para implementar la interfaz de línea de comando o CLI (por sus siglas en ingles), donde recibe un monto de parámetros y ofrece una ayuda al usuario, como muestra la figura 8, que muestra todos los parámetros que puede recibir por la entrada estándar.

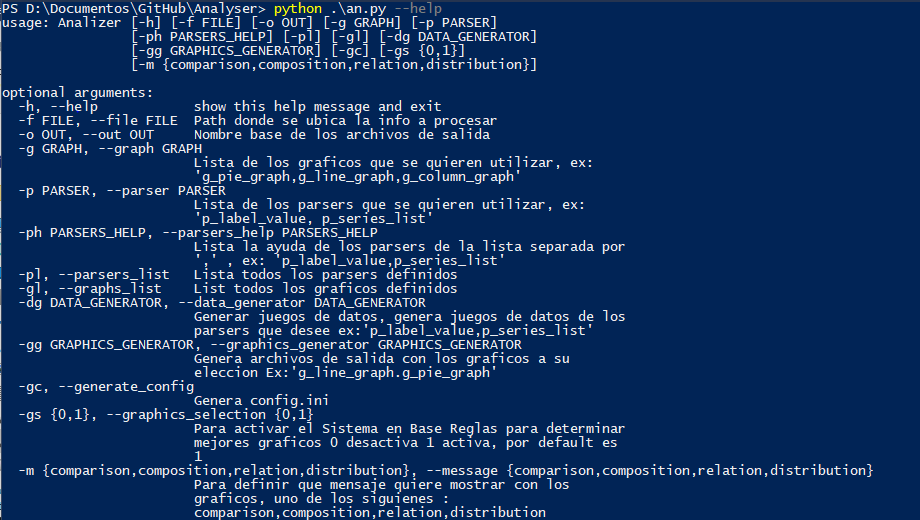


Figura 8 Ayuda del CLI.

Acorde estos parámetros la aplicación establece la configuración y pasa a mostrar ayuda, generar juegos de datos o analizar un archivo, acorde al pedido del usuario. La configuración es establecida por los parámetros de entrada y por el archivo “config.ini”, se ve un ejemplo de este file en la figura 9, este último no tiene necesidad de existir, en caso de que exista se leen sus valores y se configura la app acorde a ellos, para luego configurarla acorde a los parámetros de entrada, teniendo estos mayor prioridad. Esta configuración es guardada en una clase “Config()”.

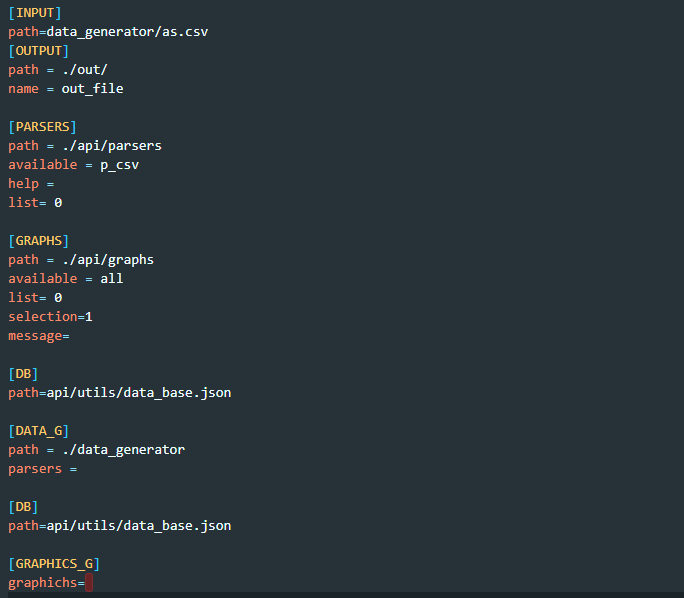


Figura 9, ejemplo de archivo “config.ini”

El archivo “config.ini” puede o no estar presente en el programa como ya se dijo. Está compuesto por una serie de campos los cuales serán utilizados para establecer la configuración, no todos los campos tiene que estar llenos, y pueden faltar campos en el archivo, esto no influye en el funcionamiento del programa, en caso de faltar alguno el programa toma su valor por defecto. En este se establece la dirección del archivo a procesar, la dirección donde será escrita la salida del programa… etc Ver si hay que hablar de todos los campos, explicándolos!!!!

La clase “Config()” es la encargada de almacenar en una serie de parámetros la configuración del programa, esta se crea en un inicio con una configuración por defecto que luego será sobrescrita por los parámetros de entrada y por el archivo “config.ini” si existe. Esta clase implementa el patrón singleton, asi solo se puede crear una instancia de la misma.

El archivo “an\_identify.py”

El archivo “An\_Known\_ Format.py” solo es una capa para hacer visible los KF, importando los formatos conocidos definidos en el directorio “known\_formats” que en estos momentos son muy simples.

El archivo “An\_Graph”

Los resultados se guardan en la carpeta output

Requerimientos:

1. python 3
2. python highcharts
3. argparse
4. electron