Tesseract OCR software.

Introducción.

Tesseract OCR es un software de reconocimiento óptico de caracteres, hecho ahora en código abierto (o como se conoce mejor, Software Libre). "OCR" es el acrónimo de tres letras (en inglés) para "Optical Character Recognition" el cual consiste en extraer texto de una imagen dada (ya sea un documento digitalizado: una foto, manuscrito, etcétera o en su defecto generado por ordenador, el cual es nuestro caso y lo haremos más adelante). Dicha tecnología existe desde finales del siglo veinte, en incluso nosotros poseíamos un "escáner" (digitalizador de documentos, artefacto parecido a una fotocopiadora) que se comunicaba via puerto paralelo al ordenador e incluso tenía otro puerto adicional para conectar a la impresora de puerto paralelo -matriz de puntos- al escáner -con un interruptor mecánico- ya que la tecnología era extremadamente costosa para la época y pocos se podían dar el lujo de agregar puerto paralelos por ranura PCI o una tarjeta USB (pero se debía adquirir también una impresora tipo USB, cara también).

En aquella época cada escáner traía su propio CD (700 megabytes) de donde uno instalaba el software y aunque existía el internet era inviable intentar descargar tal cantidad de datos por un módem de 56K. Pero hoy en día la cantidad de software es inmensa, tanto privativo o de código abierto, y Tesseract OCR es apenas uno de ellos. Podéis revisar una lista evaluada en Wikipedia pero no se limiten, el mundo es muy grande.

Es por esto que recalcamos que el mundo es inmenso: Tesseract OCR fue desarrollado por la empresa Hewlett-Packard en sus propios laboratorios radicados en Inglaterra y Estados Unidos entre 1985 y 1994 (recordad lo que os dije anteriormente sobre los limitados recursos de hardware de aquella época, por eso tantos años de desarrollo). En 1996 se logró una versión para el sistema operativo Windows y se migró -en parte- al lenguaje C++ en 1998. Poco se avanzó al llegar el siglo XXI pero en el año 2005 la empresa liberó el código fuente emplazándolo en la Universidad de Nevada (Estados Unidos) y en el año 2006 es patrocinado su mantenimiento por parte de la empresa Google. Tesseract OCR es actualmente mantenido en linea (su repositorio, código fuente), abierta para todos, por la empresa GitHub y están a cargo de dicho proyecto el señor Ray Smith acompañado de 27 colaboradores.

La necesidad es la madre de las invenciones.

La Gaceta Oficial de Venezuela (podéis leer su historia completa en nuestro blog) es la encargada de publicar las Constituciones, leyes, decretos, providencias, nombramientos y avisos oficiales (entre otras cosas) para que tengan vigencia legal en todo nuestro territorio nacional. Su importancia es capital para el funcionamiento de nuestra República Bolivariana. Sin extendernos mucho en su historia (que para ello tiene su propia entrada en esta web) del tradicional papel se ha modernizado y en un principio era simplemente digitalizada del papel impreso en imágenes en

formato gif pero de unos años para acá esas imágenes gif se estaban incrustando en formato pdf (portable document format) a fin de poderlo imprimir y llevar en un solo archivo.

El método anterior es muy seguro: una fotografía del texto impreso en papel, una fiel copia del original. Pero con la modernización de las imprentas, que dejaron de ser analógicas y ahora aceptan formatos digitales (aunque sigan usando tinta y papel) se ha decidido "hacerla" en pantalla y luego enviarla a los talleres. Hasta aquí todo bien pero se planteó el problema de publicar vía electrónica (Internet) y aquí es que se presenta el dichoso problema que aquejan a nuestros estudiantes hoy día: EL INFAME COPIAR Y PEGAR.

Es por ello que en la Imprenta Nacional decidieron echar mano del método descrito MUY BIEN en esta página web para ofuscar el documento pdf. Vamos a simplificar al máximo porque si queréis conocer todos los detalles debéis leer esa página web (y traducirla):

- Los laboratorios Xerox de Palo Alto, California (Estados Unidos) y luego posteriormente Steve Jobs, fundador de Apple Computer, inventaron y difundieron las fuentes gráficas para las computadoras (recientemente falleció el profesor de caligrafía quien dio clases a Steve Jobs en la universidad).
- Dichas fuentes son simplemente archivos de imágenes (repito simplificando al máximo) pero con un detalle importante: cada letra -en codigo ASCII (y luego en UNICODE)- está embebida en su correspondiente imagen.
- Esto se logró realizar debido al avance de las computadoras y el aumento de su potencia y velocidad con el paso de los años: son capaces de dibujar muy rápidamente en pantalla, puntito a puntito, píxel a píxel cada letra en pantalla.
- Sin embargo podéis ver aún que cuando arranca una computadora, ya sea presionando escape o alguna otra tecla dedicada observaréis el texto puro y simple del monitor. Es más rápido pasarle al monitor el código ASCII (con su ubicación en fila y columna, generalmente 80 columnas y 25 lineas) y el propio monitor en su memoria inscrustada sabe cómo dibujar la letra a o el número seis (tiene sus fuentes grabadas en "firmware").
- Pues así son las fuentes instaladas en nuestras computadoras: le dicen al ordenador cómo dibujar en pantalla (de nuevo simplificando al máximo) con ayuda del sistema operativo que se comunica con el monitor y le indica el tamaño del "lienzo" (actualmente la resolución mínima es 1024x768 píxeles) y además la ubicación de cada "letra" que vemos.
- El problema se presenta por los diversos sistemas operativos y hardware: al pasar un documento con una muy bonita fuente a otro ordenador *que no tiene instalada dicha fuente* se pierde enormemente el trabajo realizado (aunque sigue siendo legible porque el sistema operativo se encarga de sustituir por una fuente predeterminada *pero no se ve igual al original*).
- Es por ello que se inventó el Documento con Formato Portátil (Portable Document Format): en un solo archivo se incluye el código ASCII o UNICODE de cada letra del documento *además de las fuentes escogidas por el autor del documento.* Por supuesto esto abulta el tamaño del archivo pero para compensar esta desventaja aprovecharon de incluir

características adicionales: se pueden incluir imágenes, enlaces web y **lo más importante: cómo dibujar cada letra en el sitio correcto para que se vea idéntico a como originalmente se hizo.**

- Aquí viene lo interesante de la ofuscación del documento: se toma el texto original, de cada letra su código ASCII o UNICODE y se cambia según fórmula predeterminada por otro caracter ASCII o UNICODE diferente y ***a la vez también se hace lo mismo con la fuente que se incluye en el documento***.
- Es así entonces, por ejemplo, si tenemos una letra A cuyo código ASCII es 65 y la aplicamos una fórmula sencilla (sumarle 100) en el PDF colocamos el código ASCII 165 y la fuente que dibuja el caracter 195 la modificamos y ahora dibuja una letra A.
- Luego al hacer el archivo pdf pues incluimos la fuente modificada para visualizarla como "texto" normal, seleccionable y copiable, **pero que al pegar en otra ventana producirá como resultado el caracter ASCII 165 (no la letra deseada, la "A", código 65).**
- Y volvemos a repetir, esto es simplificación al máximo, volvemos a indicar el enlace donde explican la ofuscación en pdf.

Explicado "brevemente" lo anterior, observamos que las Gacetas Oficiales publicadas en Internet por la Imprenta Nacional gozan de dicha técnica **pero más complicado aún:** seleccionan párrafos enteros con una fórmula aplicada y fuente modificada pero diferentes para cada párrafo. Esto en realidad parece difícil de realizar (ya es difícil "dibujar" uno mismo sus propias fuentes tipográficas) más difícil aún modificar varias fuentes, pero echando mano del lenguaje PostScript (lenguaje orientado y basado en vectores gráficos) se puede automatizar la tarea y luego convertirlo a PDF (de nuevo: leed el artículo completo -en inglés-).

El resultado es el siguiente: en el documento PDF abierto seleccionas una porción de texto e inmediatamente se ve de esta manera (al copiar y pegar en nuestro editor de texto **gedit** solo se ven un monto de caracteres sin sentido):

Debido a esto, y como nuestro trabajo de programación debe cumplir parámetros legales (más que todo con las Providencias del SENIAT) que son publicadas en la Gaceta Oficial (amén de muchísimas otras leyes del SUNDDE, SUNAGRO, etc -por nombrar unos cuantos apenas-) nos damos a la tarea de transcribirlas y publicarlas por este vuestro humilde sitio web. Aquí es entonces que echamos mano del Reconocimiento Óptico de Caracteres (y no os califiqueís a nosotros de "flojos" sino que queremos hacer más en menos tiempo PARA HACER MÁS TRABAJO CON ESE TIEMPO AHORRADO, en serio).

Instalación de Tesseract OCR.

Probablemente en su distribución GNU/Linux ya tengáis instalado el Tesseract OCR (casi lo olvido, siempre acompaño las siglas OCR cuando lo escribo porque en realidad un "tesseract" -teseracto en castellano- es un cubo pero en la cuarta dimensión, demostrado por teorías

matemáticas pero difícil de imaginar -de allí que lo nombran en muchas películas, la más famosa "The Avengers"-). Como ya dijimos, al ser Software Libre podemos acudir a su repositorio e instalarlo nosotros mismos pero eso escapa al tema tratado en este tutorial. Si tenéis vuestros repositorios bien orientados lo único que debéis hacer es abrir una ventana terminal, ganar acceso como usuario raíz **root** e introducir la siguiente orden:

```
apt get install tesserac-ocr
```

En nuestro caso nuestro Ubuntu ya tiene instalado y nos presenta el siguiente mensaje:

Instalación de Shutter.

Otro software de código abierto que utilizaremos es Shutter el cual nos permite tomar capturas de nuestra pantalla de ordenador de una manera muy ordenada y de una forma más poderosa que la tecla de imprimir pantalla ("ImpPant" o "PrintScreen"). Dicha tecla hace muchísimos años cuando los sistemas operativos no tenían interfaz gráfica (no nos andemos con rodeos. cuando existía el CP/M y el MS-DOS) "tiraban" a la impresora de matriz de punto de puerto paralelo lo mostrado en pantalla (es decir sus códigos ASCII) y era rapidísimo porque eran unos cuantos bytes y ya la impresora "sabía" cómo imprimirlos según su memoria en "firmware". *En los sistemas operativos modernos ahora dicha opción consiste en guardar en formato de imagen lo que muestra en pantalla, ya sea en el portapapeles o en un archivo en la carpeta "imágenes" del usuario.*

https://twitter.com/ks7000/status/736698558859087872

Para instalar Shutter debemos realizar los pasos previos cuando instalamos Tesseract OCR pero esta vez cambiamos la orden de la siguiente manera:

```
apt-get install shutter
```

Si ya está instalado veréis algo similar a esto:

Instalando Shutter en Fedora y Ubuntu 18

Actualizado el día martes 11 de diciembre de 2018

En otra entrada dedicada totalmente a Shutter explicamos cómo instalarlo en Fedora y además, si

queremos utilizar sus herramientas de edición de imágenes, explicamos cómo habilitarlas, **especialmente en Ubuntu 18**, ya que dichas librerías no se encuentran en los repositorios habituales.

Configurando Shutter

Repetimos: hoy en día al uno imprimir pantalla, según sea el sistema operativo que uséis, o bien lo copia como imagen al portapapeles o bien lo guarda de una vez en la carpeta "Imágenes del usuario" tal como hace GNU/Linux Debian. Con Shutter podremos hacer muchísimas más cosas al capturar nuestra pantalla y una de ellas es poder tomar *solamente partes específicas de lo que vemos en nuestro monitor.* Para ello lo ejecutamos y al "cerrarlo" el programa se queda en la bandeja del sistema con un ícono al cual le hacemos clic derecho y al desplegar el submenú escogemos "Selección" y arrastramos con el ratón lo que deseemos capturar y luego presionamos la tecla "Intro" o "Enter".

En nuestro caso vamos a cambiar la carpeta donde vamos a guardar nuestras imágenes seleccionadas en una carpeta nueva con el nombre de la Gaceta Oficial más el número de ejemplar, esto para trabajar de manera un tanto organizada:

Si no os gusta el prefijo "Selección_" podéis cambiarlo como necesitéis y/o gustéis, acá las opciones automatizadas en variables canónicas:

Capturando imágenes para convertirlas a texto.

Pues acá un ejemplo de lo que se deseamos a convertir:

De nuevo abrimos una ventana terminal e introducimos el siguiente comando, estando ubicados en la carpeta donde guardamos las selección arriba mostrada:

```
tesseract nombre_imagen nombre_archivo.txt
```

Veriamos algo similar a esto al ejecutarlo:

Acá "copiamos y pegamos" lo que convertimos para que podamos analizarlo:

```
SUMARIO   PRESIDENCIA DE LA REPUBLICA  Decreto N° 2.323, mediante el cua
l se declara el Estado de  Excepción y de la Emergencia Económica, dadas
Ias  circunstancias extraordinarias de orden Social, Económico,   Politi
co, Natural y Ecológicas que afectan gravemente Ia  Economia NaCIonal.
```

Evidentemente que existe un margen de error al convertir la imagen; podemos contar los siguientes:

1. "Excepción".
2. "Económica".
3. "Económico".
4. "Ecológicas".
5. "NaCIonal".

Cinco errores en cuarenta y un palabras, aproximadamente un 12% de margen de error. **Sin embargo, a la larga, resulta menos "trabajoso" que tipear letra por letra, es decir, no tiene tanto trabajo corregir el texto convertido.**

Actualizado el viernes 22 de septiembre de 2017.

Más vale tarde que nunca, acá colocamos el enlace al "Hunspell Tutorial" que permite de manera automática -en sumo grado- el corregir estas palabras mal digitalizadas, esperamos le sean de utilidad en su trabajo diario.

Trabajo de proceso por lotes.

Con un solo archivo ya vimos lo que nos espera, el panorama en general. *El problema es que tenemos 43 archivos y debemos tipear (y modificar los números) de 43 lineas de comandos.* ***Mucho trabajo, y lo que queremos es hacer más en menos tiempo (y con menos esfuerzo).***

Así que echaremos mano de la programación de comando BASH de la siguiente manera:

- Listamos los archivos deseados según un patrón predeterminado.
- Dicho listado lo guardamos en un archivo de texto.
- Luego leemos, linea por linea, dicho archivo de texto y lo guardamos en una variable de cadena.
- Hacemos una linea de comando sustituyendo el nombre del archivo con la variable leída.
- Al final concatenamos en un solo archivo, **editamos los errores que encontremos simplemente leyendo y corrigiendo** y ya podremos "copiar y pegar" hacia donde lo necesitemos.

El archivo de comandos por lotes tendrá estas características (comentario explicativos incluidos, son las lineas que comienza con numeral "#"):

```
#!/bin/sh ####Licencia de uso### # Copyright 2016 Jimmy Olano at ks7000.net.ve # # Licensed under the Apache License, Version 2.0 (the "License"); # you may not use this file except in compliance with the License. # You may obtain a copy of the License at # # http://www.apache.org/licenses/LICENSE-2.0 # # Unless required by applicable law or agreed to in writing, software # distributed under the License is distributed on an "AS IS" BASIS, # WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. # See the License for the specific language governing permissions and # limitations under the License. ###################### # Cambiar la siguiente variable por el prefijo deseado patron="Selección_*" # Cambiar la siguiente variable para el nombre del archivo deseado nomarch="lista.txt" # Limpiamos pantalla clear # Guardamos en un archivo de texto los nombres de los archivos a convertir ls $patron > $nomarch # Creamos un archivo de texto vacio para inicializar ciclo touch documento.txt # Leemos linea por linea el archivo lista.txt while read linea do # Enunciamos el archivo con que vamos a trabajar (los seres humanos somos ansiosos) echo "Procesando "$linea # Aqui colocamos la orden de convertir a texto (automaticamente agrega extension txt) tesseract $linea $linea # Concatenamos el archivo recien convertido en un archivo temporal cat documento.txt $linea".txt" > documentotmp.txt # Renombramos el archivo temporal a su nombre original mv documentotmp.txt documento.txt # Borramos el archivo de texto convertido luego de haberlo integrado al documento completo rm $linea".txt" done
```