UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería en Informática

TFG del Grado en Ingeniería Informática

título del TFG
Documentación Técnica

Presentado por nombre alumno
en Universidad de Burgos — 15 de abril de 2017
Tutor: nombre tutor

Índice general

Índice de figuras

Índice de tablas

Apéndice A

Plan de Proyecto Software

A.1. Introducción

A.2. Planificación temporal

Primer Sprint: Este tramo inicial fue dedicado a realizar investigación sobre las herramientas que se usarían para desarrollar el producto final. Se buscó información sobre 3 elementos o partes distintas: • Herramienta para desarrollar la GUI: Concretamente, se debatió acerca de si sería más adecuado crear una aplicación web o una aplicación interna. Para tomar una decisión, se tuvieron en cuenta 4 herramientas. Por la parte de aplicación web, las candidatas fueron Jupyter y Flask; y por la parte de aplicación de escritorio, PyQt y Kivy. Tras practicar con ambas herramientas, se tomó la decisión de programar la GUI en un notebook de Jupyter para posteriormente publicarlo en un Dashboard. • Base de datos para almacenar la información: Como se debía almacenar numerosa y variada información sobre las noticias elegidas, se necesitaba de una estructura donde almacenar todos esos datos. La decisión más relevante en este apartado era si usar una base de datos relacional o no relacional, o dicho de otra manera, SQL o NoSQL. Las alternativas fueron AlchemySql (relacional), PyLucene (no relacional) y PyMongo (no relacional). • Herramienta para escribir la documentación: Aquí sólo se plantearon dos alternativas inicialmente: OpenOffice o Latex. Por cuestiones de formato y de preferencia de los miembros del proyecto, se optó por Latex, aprovechando la plantilla para Trabajos de Fin de Grado que ya había sido creada y que estaba a disposición de los alumnos.

Segundo Sprint: La prioridad de este sprint era finalizar la investigación y búsqueda de herramientas para desarrollar la aplicación, que concluyó con la elección de Pymongo como base de datos, y los notebook de Jupyter como herramienta de interfaz. Una vez elegidas las diferentes herramientas nece-

sarias para el desarrollo del proyecto, el siguiente paso consistió en crear un pequeño programa que probara la librería PDFMiner y que contuviera algo de funcionalidad básica como contar la frecuencia de una palabra dentro de un documento. Aprovechando este script, se realizaron pruebas con un snippet llamado fileupload, que permite cargar ficheros en el código usando una GUI clásica en la que puedes navegar por el equipo buscando la ubicación del mencionado fichero. El resultado final era un programa que permite al usuario buscar un fichero en el equipo y, si este archivo tiene una extensión pdf, se permitirá buscar la frecuencia de las palabras que introduzca el propio usuario.

Tercer Sprint: Habiendo finalizado la elección de todas las herramientas necesarias, y probado el funcionamiento básico para lectura de PDF subiendo archivos a mano, la siguiente prioridad fue crear un primer prototipo de web scrapping. Para lograr esto, se dividió la funcionalidad en dos etapas: • Un primer script que realiza la lectura web de noticias en diferentes medios (para este prototipo, se leían noticias sólo del diario "Público" y, más en concreto, de la sección de Igualdad), almacenando en la base de datos estructuras formadas por el titular, autor, fecha, texto de la noticia, procedencia y un link a la misma. Para recoger el texto html de las páginas web de los diarios, se usaron las librerías de url2lib y BeautifulSoup; la primera para hacer una llamada a la url indicada y obtener todo (o casi todo, en las próximas etapas explicaremos qué faltaba) el texto HTML de la página de cada noticia, y la segunda para poder manipular todo el HTML recibido previamente de forma más sencilla sin tener que hacer una división manual por etiquetas, simplemente indicamos al contenido de qué etiquetas queremos acceder.

• Un segundo script que, en función de una palabra clave introducida por el propio usuario, consulte en la base de datos y extraiga todas las noticias que contengan esa palabra clave, en un formato de tabla HTML y mostrando la información más importante de la noticia junto con el número de apariciones que hace esa palabra en el texto de las mismas. Para poder mostrar esto en la salida del notebook, necesitamos las librerías de pandas, ipywidgets y qgrid. Con la librería de pandas podemos organizar la información devuelta en la consulta de la base de datos a un dataframe, que puede ser convertido a una tabla html. Con los widgets, podemos mostrar controles básicos como botones, entradas de texto, listas desplegables, etc... sin necesidad de escribir a mano el código HTML. En esta versión, se usó la librería qgrid para transformar el dataframe en una tabla interactiva, que permitía reordenar el orden de las filas dando prioridad a unas columnas u otras. No obstante, no permitían formatear las celdas por separado, lo cual nos hizo descartarlo para posteriores versiones.

Cuarto Sprint: Habiendo realizado una primera aproximación tanto a la lectura de PDF como de web, este sprint se centró en mejorar tres apartados: • En la parte de PDF, se mejoró el script, aunque a costa de retirar tempo-

ralmente la librería que permitía seleccionar un fichero de forma manual, para que recorriera y almacenara todos los archivos PDF, tanto del directorio que se pasaba como argumento como de todos los que colgaban de él. Una vez recorridos todos, se guardan las noticias en la base de datos, de forma similar a como se hacía con las noticias en la web pero cambiando el atributo de la procedencia. • En la parte de web scrapping, se actualizaron dos aspectos importantes: o Por una parte, usando Selenium además de BeautifulSoup se consiguió obtener los comentarios de las noticias, que en el caso de "Público", se generaban con una API de terceros que impedía acceder a su HTML de forma normal como se hacía con el resto de la noticia. El método era algo pesado, ya que selenium necesita abrir en el ordenador que se esté ejecutando las pestañas a las que accede, pero como ya se ha mencionado, fue la alternativa que se encontró más idónea para poder acceder al HTML de los comentarios de las noticias. o Por otro lado, se cambió el formato de la tabla, dejando de usar qgrid para el DataFrame y mostrándola en un HTML normal, al que luego se aplicaba estilo manualmente usando un script de Javascript. Ésto es posible ya que uno de los widgets de Ipython/Jupyter permite introducir código HTML de forma manual, y es la vía que también utilizamos para introducir código JavaScript dentro del output del notebook. Con este cambio, tenemos flexibilidad total para manipular las celdas de la tabla para, por ejemplo, formatear como hiperenlaces las url de las noticias que se muestran en la tabla.

Quinto Sprint: En esta etapa se decidió que en el muestreo de resultados, sería interesante incluir gráficos que acompañaran al resto de la información, para poder mostrar la mayor cantidad de datos posibles de diversas formas. Por tanto, una de las tareas fue elegir una librería de gráficos adecuada a nuestro contexto. Las candidatas iniciales fueron Bokeh y SeaBorn. Se probó Bokeh y, debido a unos problemas relacionados con la generación de archivos HTML, de decidió descartar. Por falta de horas durante esa semana, se trasladaron las pruebas con SeaBorn al siguiente sprint. La otra gran tarea a la que se dedicó tiempo durante este sprint fue a aumentar los medios a los que se realizaba web scrapping, creando un script para recorrer y almacenar las noticias de ElDiario. En este caso, los comentarios no se generaban a partir de una API de terceros, pero debido a la complejidad del código HTML obtenido, y visto necesario el uso de Selenium una vez más, se decidió separar la obtención de comentarios y aplazar esa parte al siguiente sprint.

Sexto Sprint: La mayor parte de esta semana se dedicó a terminar tareas de la semana anterior que habían sido subestimadas en horas, y separadas en tareas entre este sprint y el anterior. En el apartado de los gráficos, se realizaron pruebas con la otra librería candidata, Seaborn, y por su comodidad y sencillez para mostrar gráficos simples (Como gráficos de barras para evaluar las menciones a una palabra por autor o día de la semana), se decidió que por el momento sería la alternativa elegida para acompañar en la interfaz. Respecto al web scrapping de ElDiario, se implementó el código necesario junto con el

uso de Selenium adecuado para poder obtener los comentarios de las noticias de ese medio, que era la funcionalidad que no dio tiempo a terminar en la anterior semana. Aparte de estas tareas, se dedicó cierta cantidad de tiempo a investigar formas de publicar esta aplicación web en algún servidor. Para comenzar, se realizó cierta investigación sobre Docker, además de ver vídeos de la PyCon de Londres 2016, donde se trataba el tema de subir notebook a un servidor y poder visualizarlos como un dashboard. Por problemas de límite de tiempo, no se pudo avanzar mucho más este aspecto del proyecto.

Séptimo Sprint: En este sprint se dedicó parte del tiempo a solucionar un problema recurrente en el que se mostraban sucesivamente todas las tablas de resultados de todas las palabras buscadas, provocando que la interfaz resultante se llenara de tablas innecesarias. De la misma manera, con los gráficos de Seaborn, que usa MatplotLib para funcionar, sucedía algo similar, provocando que todos los gráficos generados se fueran acumulando en una misma zona de dibujado que comparten por requisitos de la librería. Para solucionar esto hicieron falta dos medidas distintas. Para la parte de las tablas html, el problema se solucionó desde el código de Python, cambiando el planteamiento de la función que se ejecutaba al pulsar en el botón de búsqueda, y haciendo globales algunas variables necesarias para la generación de estas tablas. Para la parte de los gráficos, como este problema se estaba produciendo por limitaciones de la librería, hubo que buscar una solución más "rudimentaria" y controlar, por medio de una función JavaScript (introducida en un widget HTML), la visualización de la tabla requerida y la no visualización del resto de tablas obsoletas. El resto del tiempo de esa semana se dedicó a la investigación de la herramienta TextBlob, una opción muy interesante para realizar minería de datos de texto. En términos generales, es una herramienta que, a partir de los textos recibidos, puede traducirles, detectar idiomas, corregir palabras, dividir párrafos en frases, y frases en palabras, devolver etiquetas de las frases o palabras... etc. Aparte de toda esta funcionalidad, muy útil para casos como el de este proyecto, en el que se manejan textos, tiene sus propios clasificadores de datos. Para usar esta herramienta, se necesita descargar el NLTK (Natural Language ToolKit), que contiene todos los conjuntos de datos necesarios para poder llevar a cabo estas manipulaciones de texto y clasificaciones de los mismos. Debido a problemas con la descarga y tamaño de los archivos, se cerró la tarea en ese punto, y se retomó en el siguiente sprint. Al término de esta semana, se realizó una reunión con el cliente para enseñar los progresos alcanzados hasta ese punto y tomar decisiones acerca de qué pasos tomar a continuación.

Sprint 8: La tarea más importante realizada esa semana fue la de hacer otra aproximación al web scrapping basada en obtener los datos a partir de los RSS de los medios en los que, hasta el momento, se estaban leyendo desde las páginas web convencionales. El resultado fue un pequeño programa que te permitía seleccionar un medio de una lista de diarios y una palabra clave y,

en tiempo real, realizar el web scrapping del RSS de ese medio y almacenarlo en la base de datos a la vez que se mostraba en una tabla los resultados obtenidos. Para ello se usó la librería feedparser. Este acercamiento era mucho más rápido que el web scrapping convencional, pero a cambio sacrifica la posibilidad de poder acceder a los comentarios de las noticias, y los casos de algunos medios, no todo el texto de la noticia es accesible a través del RSS. Como ha sido mencionado en el resumen del sprint anterior, debido a la falta de capacidad en la máquina virtual en la que se estaba trabajando, no se podía descargar todo el conjunto de datos de NLTK de forma correcta, lo cual se intentó solucionar inicialmente clonando la máquina virtual a una de almacenamiento dinámico y editando la capacidad de almacenamiento de las particiones de la misma. Esta solución no tuvo resultados positivos, por tanto, se reinstaló desde el principio una nueva máquina virtual que evitara estos problemas y que además corrigiera la instalación de algunas librerías que habían producido problemas previamente. Por último, se probó una nueva librería de gráficos que sustituyera a Seaborn, denominada Bqplot, la cual se asemejaba en funcionalidad a Bokeh, pero evitando esa necesidad de archivos HTML que nos habían llevado a descartar Bokeh en un primer momento. Las pruebas fueron muy positivas en gráficos de barras y lineales.

Noveno Sprint: La duración de este sprint fue el doble de lo normal (dos semanas en vez de una) debido a que coincidió con las vacaciones de Semana Santa. Por tanto, el día que debía haberse realizado la reunión habitual no era lectivo, y hubo que posponerlo. Por otro lado, como había suficientes tareas/horas de trabajo como para dividirlo en más de un sprint, se dio por finalizado el sprint a los 14 días aunque la siguiente reunión fuera después de 21. En lo relacionado a tareas relacionadas con sprints anteriores, se consiguió solucionar el problema por el cual no se aplicaban los scripts de JavaScript, aunque para ello hubo que buscar una forma alternativa de implementarles. Además, se consiguió finalizar con éxito la instalación y pruebas con TextBlob, llevando a cabo la instalación del NLTK de una manera alternativa que ocupaba mucho menos espacio en memoria. Una vez instalado y probado, se hizo un sencillo prototipo con un conjunto de datos de prueba minúsculo de diferenciador de frases machistas y feministas, tanto con TextBlob como con SciKitLearn, aprovechando código de scripts ya existentes. También se mejoraron las operaciones de almacenamiento en base de datos, evitando introducir noticias duplicadas, para que el usuario final no se tenga que preocupar de filtrar las que estén duplicadas de las que no. En lo respectivo a la publicación del notebook como aplicación web, se realizaron instalaciones de varias herramientas de Jupyter Dashboards y de Docker, pero debido a impedimentos de configuración de la BIOS y del anfitrión, no se puedo avanzar más en la máquina virtual, y se decidió repetir el proceso en el anfitrión en el siguiente sprint. Por último, se añadió el diario "El Mundo" a los disponibles en el web scrapping por RSS, para ir aumentando el rango ideológico de las alternativas

disponibles.

A.3. Estudio de viabilidad

Viabilidad económica

Viabilidad legal

Apéndice B

Especificación de Requisitos

B.1. Introducción

B.2. Objetivos generales

B.3. Catalogo de requisitos

B.4. Especificación de requisitos

Apéndice C

Especificación de diseño

C.1. Introducción

C.2. Diseño de datos

C.3. Diseño procedimental

C.4. Diseño arquitectónico

Apéndice D

Documentación técnica de programación

Apéndice E

Documentación de usuario

E.1. Introducción

E.2. Requisitos de usuarios

E.3. Instalación

E.4. Manual del usuario