UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería en Informática

TFG del Grado en Ingeniería Informática

Framework web de uso de sistemas de machine learning.

Presentado por Javier Martínez Riberas
en Universidad de Burgos — 14 de febrero de 2017
Tutor: Dr. José Francisco Díez Pastor y Dr. César Ignacio García Osorio

UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería en Informática

D. José Francisco Díez Pastor y D. César Ignacio García Osorio, profesores del departamento de Ingeniería Civil, área de Lenguajes y Sistemas Informáticos.

Expone:

Que el alumno D. Javier Martínez Riberas, con DNI 71299495R, ha realizado el Trabajo final de Grado en Ingeniería Informática titulado ” .

Y que dicho trabajo ha sido realizado por el alumno bajo la dirección del que suscribe, en virtud de lo cual se autoriza su presentación y defensa.

En Burgos, 14 de febrero de 2017

Vº. Bº. del Tutor:	Vº. Bº. del co-tutor:
D. José Francisco Díez Pastor	D. César Ignacio García Osorio

Resumen

Investigar es dificil -¿investigar redes neuronales tambien -¿se busca agilizar el proceso de investigación y estudio de nuevas tecnologias. En concreto se busca el poder obtener datos(serian residuales) a los cuales se les pueda aplicar mineria.

Descriptores

Mineria de datos, redes neuronales, clasificadores, aplicación web. . . .

Abstract

A **brief** presentation of the topic addressed in the project.

Keywords

keywords separated by commas.

Índice general

Índice de figuras

Índice de tablas

Introducción

Descripción del contenido del trabajo y del estrucutra de la memoria y del resto de materiales entregados.

Objetivos del proyecto

A continuación se indican los objetivos, tanto teóricos marcados por los requisitos como objetivos que se persiguen con el proyecto. También se incluyen ciertos requisitos que no tienen por que ser obvios pero que en la actualidad se esperan de cualquier aplicación web.

2.1. Objetivos principales

- Facilitar investigacion y explotación de sistemas de machine learning.
- Unificar una interfaz para interactuar con sistemas de machine learning.
- Posibilitar el uso de las características anteriores mediante una aplicación web.

2.2. Objetivos parte mineria

- Facilidad de incremento en el numero de sistemas de machine learning
- Flexibilidad en los posibles sistemas de machine learning
- Flexibilidad en el origen de los sistemas de machine learning
- Extraccion de datos desechables para investigacion
- Flexibilidad en los datasets introcucibles

2.3. Objetivos parte web app

- Arquitectura MVC (Model View Controller)

- Facilidad usable
- Internacionalizacion posible
- Responsiveness

2.4. Objetivos proceso de desarrollo

- Usar algún sistema para el control de dependencias y despliegue de el proyecto y dependencias
- Utilizar un sistema de control de versiones
- Tener un proceso de CI (Continuous integration)
- Programar con agilidad, no agilmente
- Programar siguiendo el Manifesto for Software Craftsmanship
- Distribuir el proyecto con alguna herramienta de fácil uso

2.5. Objetivos técnicos en cuanto al proceso de desarrollo

- Para el control de dependencias y depliegue del proyecto y de las mismas se usará pip (freeze e install -r)
- Como sistema de control de versiones se utilizará git y github
- El proceso de CI se facilitará con travis y codeclimate
- Para ayudar con el feedback (retroalimentación) y el desarrollo continuo se usará Zenhub sobre github y slack como herramienta para mantener contacto con los 'clientes' (en este caso los tutores)

2.6. Objetivos personales

- Crear una herramienta que facilite lineas de investigacion poco exploradas
- Avanzar mis conocimientos a partir de los obtenidos en la carrera, sobre todo en los campos cuya relacción importancia real, conocimiento obtenido es mayor
- Profundizar en el entorno de Python, ya que en los ultimos años se ha puesto de moda.

Conceptos teóricos

Algunos conceptos teóricos tanto de la parte técnica como de la parte de procesos de software

3.1. DevOps

DevOps es un acrónimo inglés de: 'software **Dev**elopment and information technology **Op**eration**s**' es un termino que engloba un conjunto de prácticas de colaboración y comunicación entre desarrolladores software y técnicos informáticos. Los objetivos de esta comunicación y colaboración son una construcción de software más consistente y confiable. Este proceso heredero de las técnicas ágiles se basa en una *cadena de herramientas*. Esta cadena de herramientas es algo que no esta completamente definida pero más o menos la podemos concretar, cabe tener en cuenta que esta cadena cambia según a quien le preguntes.

La 'cadena de herramientas' de DevOps

Esta cadena de herramientas se basa en siete procesos con sus correspondientes herramientas:

1. Plan: Consistente en determinación de metricas, requerimientos... y una vez pasemos de la primera iteración ha de tener en cuenta el feedback del cliente.
2. Creación: Es el proceso de programar y crear el software, las herramientas en este proceso es el software de control de versiones que vayamos a usar.
3. Verificación: Proceso de comprobación de la calidad del software. Normalmente consiste en hacer test de diversos tipos (Aceptación, seguridad...)

4. Preproducción o empaquetación: En esta fase se piden aprobaciones de los distintos equipos y se configura el paquete.

5. Lanzamiento: En este punto se prepara el horario de lanzamiento y se orquesta el software para poder ponerlo en el entorno de producción objetivo.

6. Configuración: Una vez el software esta desplegado toda la parte de la infraestructura y configuración de la misma se incluye en esta categoría, como por ejemplo las bases de datos, configuración de las mismas...

7. Monitoreo: Tras entregar el software se mide su rendimiento en la infraestructura objetivo y se mide la satisfacción del usuario final. Se recogen metricas y estadísticas

Técnicas y herramientas

En esta parte de la memoria se detallan tanto técnicas y herramientas como metodologías y librerías usadas.

4.1. Metodología

Al no haber una metodología concreta que se adapte al trabajo de una sola persona se ha buscado un conjunto de metodologías que se pudiesen conjuntar para dar una metodología conjunta más aplicable a la situación particular que se tiene.

De **DevOps** se ha extraido la toolchain para minimizar riesgos totales del proyecto y se ha buscado reducir la deuda técnica nacida de la necesidad de desplegar el producto y no estar preparado para ello, algo que resulta ser más habitual de lo que debería ser.

De **Scrum** recogemos las herramientas como los sprint y las reuniones cada sprint tanto para concretar el siguiente sprint como para hacer una retrospectiva del anterior. Se usa también mucha de la documentación que propone (Burndown...).

Se ha estudiado también **extreme programming** (XP) y se ha visto que no es viable para el proyecto por la cantidad de esfuerzo que requiere para una sola persona hacer el trabajo de dos como es el pair programming.

De todos los metodos ágiles se ha adquirido la mentalidad de no intentar planificar todo desde el principio e intentar planear las cosas con el conocimiento adquirido en cada iteración. Se intenta decidir en todo lo más tarde posible para intentar conseguir el máximo conocimiento posible para mejorar las decisiones.

Se ha decidido que la recopilación de información de los sprints y este tipo de documentación se va a hacer en un tablero **kanban**, la herramienta que

nos proporciona este sistema va a ser **Zenhub**, se usa por la facilidad y el nivel de integración que nos da con otros sistemas también elegidos.

4.2. Patrones de diseño

Se ha intentado usar los patrones de diseño para conseguir una arquitectura mejor y más simple para nuevos desarrolladores. También hay que tener en cuenta que no todos los patrones tienen sentido en un lenguaje de programación como python.

Model-View-Controller

El patrón modelo vista controlador nos ayuda a simplificar mucho la forma de una página web o aplicación que dependa de vistas.

4.3. Control de versiones

El control de versiones es una necesidad hoy en día tanto en sistemas compuestos de uno o multiples desarrolladores, esto nos permite poder dejar ciertas features que empezemos a implementar y se dificulten sin terminar para seguir con otras que den más valor al cliente y luego continuar con las dificultades o dejarlas de lado según beneficie al cliente.

Este solo es un ejemplo de las ventajas entre las que se cuentan: recuperación de versiones estables, visualización de los cambios que han dado resultado a un bug...

El sistema de hosting lo usaremos para facilitar el acceso a la información contenida en el sistema, también, al ser externo (no estar en el mismo ordenador que se use) nos servirá de sistema de copias de seguridad.

Usaremos **git** por razones de documentación y conocimiento ya adquirido, aunque existen otros sistemas como subversion.

Las alternativas principales de sistema de hosteo de git son: Bitbucket, Github, Gitlab.

Se ha elegido Github por una mezcla de razones historicas con razones de integración con otros sistemas como el sistema kanban (Zenhub).

Historicamente se conocía Github y se sabe que los proyectos open source no tienen ningúna limitación, la fácil integración con otros sistemas como travis y slack si hiciese falta.

4.4. Integración continua

La integración continua es el metodo por el cuál intentamos integrar todos nuestros productos continuamente para ver si funcionan correctamente en conjunto. Nos va a ayudar a minimizar el riesgo de fallo del proyecto sobre todo si se mantiene en desarrollo un largo periodo de tiempo.

La herramienta que se va a usar es **travis**, esto se debe a la gran documentación, facilidad de integración con git y gihub, y otras integraciones que nos ayudaran con otras secciones. Otra ventaja es que al ejecutar tu build y tests en sus servidores no te tienes que preocupar de casi nada.

Otras opciones que se podrían usar son Jenkins que como ventaja es Open Source y tiene gran cantidad de plugins.

4.5. Quality Assurance

Dado que no tenemos departamento de QA como deberíamos para llevar una toolchain como la especificada en DevOps usaremos herramientas automatizadas, que aunque no tengan la calidad de una revisión humana es lo mejor que disponemos.

La herramienta que se ha elegido es **CodeClimate**, se ha usado por que no ha sido demasiado complicado el incluir esta herramienta en la linea de producción que ya teníamos (Github+travis). Se ha intentado usar SonarQube pero no ha sido tan sencillo.

4.6. Tests

Los tests se han realizado con **pytest**. Esta herramienta es muy parecida a unittest (la herramienta de tests en la biblioteca standard de python). Tiene ventajas, facilita el debug y los metodos como setup y teardown. Se integró esta herramienta con travis.

Otras librerías pueden ser unittest o nose2.

Recubrimiento

La forma más sencilla para ver el recubrimiento de un test en Python es coverage, tiene varias opciones como report para salida en consola o html para un html con el cubrimiento bien señalado. Se consiguió integrar esta herramienta con travis, pytest y CodeClimate de manera que se ejecuta con la integración continua.

4.7. Dependencias

Para el control de dependencias (seguridad, ultimás versiones y licencias) se ha usado VersionEye, integrado con Github, esta herramienta se adhiere mediante un webhook a Github y nos dice si se ha descubierto alguna brecha de seguridad en nuestras herramientas, cuales son sus ultimás versiones y si las estamos usando y si las licencias del proyecto son compatibles con la que tenemos.

4.8. Comunicación

La comunicación se ha hecho de diferentes maneras: Email, Slack y de manera física. Email se ha usado para casi toda la comunicación a distancia. Slack se ha usado para integrar herramientas y notificar sobre estas. La comunicación física es el medio que más se ha usado, esto no se debe a ningún motivo en particular pero las cosas han sucedido así.

4.9. Documentación

La documentación se ha basado en dos sistemas: en código y fuera de código.

En código

En el código se ha usado la documentación recomendada por la comunidad de python en el PEP 287, este recomienda usar **reStructuredText** (rst) para las docstrings.

Se han seguido las guidelines de pocoo, esto es para poder usar Sphinx (generador de documentación) para transformar esas cadenas de documentación en una documentación tanto en html o en Latex si hiciese falta.

Fuera de código

Para la documentación que se esta leyendo se ha usado la plantilla oficial de la ubu para crear documentos Latex aunque se ha considerado usar rst al igual que en el código ya que si hiciese falta se podría transformar a Latex o a html siendo mucho más flexible. No se ha hecho por que no existe una plantilla y la barrera de entrada es algo alta.

4.10. Editor de texto

Se ha mirado tanto en editores de texto más simples como vim, atom, sublime... Estos cuentan con suficientes plugins que acaban siendo una IDE

Las IDEs también se han mirado y se han considerado tanto Spyder, Lyclipse como Pycharm.

De todos estos no hay ventajas en usar unos u otros ya que todos acaban teniendo la misma capacidad gracias a plugins, se ha decidido usar Pycharm por que era la única IDE que todavía no se había probado.

4.11. Bibliotecas

Se ha usado Flask como microframework ya que sirve tanto para implementar el patrón MVC tanto como crear una Restfull API.

Para facilitar ciertas funcionalidades como control de usuarios, seguridad... se han usado las extensiones: flask-babel y babel(internacionalización), flask-login (control de usuarios), flask-sqlalchemy y psycopg2(acceso a la base de datos), flask-wtf y WTForms (Formularios html) y flask-bcrypt y Bcrypt para seguridad.

Aspectos relevantes del desarrollo del proyecto

Este apartado pretende recoger los aspectos más interesantes del desarrollo del proyecto, comentados por los autores del mismo. Debe incluir desde la exposición del ciclo de vida utilizado, hasta los detalles de mayor relevancia de las fases de análisis, diseño e implementación. Se busca que no sea una mera operación de copiar y pegar diagramas y extractos del código fuente, sino que realmente se justifiquen los caminos de solución que se han tomado, especialmente aquellos que no sean triviales. Puede ser el lugar más adecuado para documentar los aspectos más interesantes del diseño y de la implementación, con un mayor hincapié en aspectos tales como el tipo de arquitectura elegido, los índices de las tablas de la base de datos, normalización y desnormalización, distribución en ficheros3, reglas de negocio dentro de las bases de datos (EDVHV GH GDWRV DFWLYDV), aspectos de desarrollo relacionados con el WWW... Este apartado, debe convertirse en el resumen de la experiencia práctica del proyecto, y por sí mismo justifica que la memoria se convierta en un documento útil, fuente de referencia para los autores, los tutores y futuros alumnos.

Trabajos relacionados

Este apartado sería parecido a un estado del arte de una tesis o tesina. En un trabajo final grado no parece obligada su presencia, aunque se puede dejar a juicio del tutor el incluir un pequeño resumen comentado de los trabajos y proyectos ya realizados en el campo del proyecto en curso.

Conclusiones y Líneas de trabajo futuras

Todo proyecto debe incluir las conclusiones que se derivan de su desarrollo. Éstas pueden ser de diferente índole, dependiendo de la tipología del proyecto, pero normalmente van a estar presentes un conjunto de conclusiones relacionadas con los resultados del proyecto y un conjunto de conclusiones técnicas. Además, resulta muy útil realizar un informe crítico indicando cómo se puede mejorar el proyecto, o cómo se puede continuar trabajando en la línea del proyecto realizado.

Bibliografía