UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática

TFG del Grado en Ingeniería Informática

Visor de espectros

Presentado por Iván Iglesias Cuesta
en Universidad de Burgos — 25 de junio de 2018
Tutor: Dr. José Francisco Díez Pastor
Cotutor: Dr. César Ignacio García Osorio

UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería 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. Iván Iglesias Cuesta, con DNI 45573756S, ha realizado el Trabajo final de Grado en Ingeniería Informática titulado "Visor de espectros".

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, 25 de junio de 2018

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

Resumen

La espectroscopia Raman es una técnica de análisis no destructivo usada para conocer la estructura y composición de un material o elemento. En el campo de la geología el uso de esta técnica es común para determinar la composición, origen o profundidad de muestras de minerales extraídos.

Actualmente se están empezando a usar técnicas de minería de datos para la identificación de estos espectros. Aunque existen herramientas que son capaces de visualizar y aplicar ciertas operaciones sobre los espectros, no existe un software específico que facilite la aplicación de técnicas de minería de datos sobre los espectros.

Este proyecto se realiza en colaboración con una investigadora en geología que usa esta técnica para el análisis de un mineral en concreto llamado variscita. Este proyecto parte de unas técnicas y algoritmos desarrollados en colaboraciones previas.

El desarrollo de este proyecto busca desarrollar una aplicación web que permita cargar los espectros, visualizarlos y procesarlos, así como aplicar técnicas de minería de datos para construir modelos de clasificación para nuevas muestras.

Descriptores

Aprendizaje automático, minería de datos, procesamiento de espectros, variscita, espectroscopia Raman, aplicación web.

Abstract

Raman spectroscopy is a non-destructive analysis technique used to know the structure and composition of a material or element. In the field of geology the use of this technique is common to determine the composition, origin or depth of extracted mineral samples.

Currently data mining techniques are beginning to be used for the identification of these spectra. Although there are tools that are able to visualize and apply certain operations on the spectra, there is no specific software that facilitates the application of data mining techniques on the spectra.

This project is carried out in collaboration with a researcher in geology who uses this technique for the analysis of a particular mineral called variscite. This project is based on techniques and algorithms developed in previous collaborations.

The development of this project seeks to develop a web application that allows to load the spectra, visualize and process them, as well as applying data mining techniques to build classification models for new samples.

Keywords

Machine learning, data mining, spectra processing, variscite, Raman espectroscopy, web application.

Índice general

Índice general III

Índice de figuras V

Índice de tablas VI

Introducción 1
1.1. Materiales entregados 2

Objetivos del proyecto 3

Conceptos teóricos 5
3.1. Espectroscopia Raman 5
3.2. Visualización de datos 6
3.3. Minería de datos 6
3.4. Base de datos NoSQL 9

Técnicas y herramientas 11
4.1. Técnicas 11
4.2. Control de versiones 11
4.3. Herramientas 12
4.4. Almacenamiento 12
4.5. Librerías 12
4.6. Librerías de representación 12
4.7. Infraestructura 14
4.8. Despliegue 15

Aspectos relevantes del desarrollo del proyecto 19

5.1. Elección del proyecto 19
5.2. Formación 19
5.3. Sistema de usuarios 20
5.4. Cohesión entre Flask y Dash 21
5.5. Subida de ficheros 21
5.6. Interfaz del visor 22
5.7. Almacenamiento de datos 23
5.8. Integración de los algoritmos existentes 25
5.9. Despliegue 25
5.10. Creación de clasificadores 27

Trabajos relacionados **29**
6.1. Artículos científico 29
6.2. Software 29
6.3. Librerías 30

Conclusiones y Líneas de trabajo futuras **31**
7.1. Conclusiones 31
7.2. Líneas de trabajo futuras 32

Bibliografía **35**

Índice de figuras

3.1. Representración de la dispersión de fotones[9] 5
3.2. Recorte entre valores 200 y 1500 7
3.3. Corrección de la línea base 8
3.4. Normalización de los datos 8
3.5. Compresión de los datos 8
3.6. Suavizado con ventana de tamaño 25 9
5.7. Tabla de espectros 23
5.8. Estructura definitiva 24

Índice de tablas

Introducción

La espectroscopia Raman es una técnica espectroscópica basada en la dispersión inelástica de fotones, o dispersión Raman, de la luz monocromática disparada comúnmente desde un láser[17], generalmente en el rango de la luz visible o cercano al infrarrojo o ultravioleta[24]. Al medir la energía de estos fotones dispersados se obtienen espectros que revelan información estructural sobre el elemento o material. Esta técnica se explica más en profundidad en la sección 3.1.

En el campo de la geología esta técnica es muy útil debido a la complejidad en la estructura de las rocas, generalmente formadas por varios tipos de minerales, que esta técnica es capaz de averiguar. Además permite obtener información en profundidad sobre su formación debido a que la espectroscopia Raman es muy sensible al mínimo cambio en la estructura del material.[12].

En el área de lenguajes y sistemas informáticos existen colaboraciones en curso entre el grupo de investigación ADMIRABLE[1] y la geóloga Susana Jorge Villar, investigadora de la UBU actualmente adscrita al CENIEH[2] en un programa de investigación en geoarqueología[6], aunque es experta en espectroscopia Raman aplicada a astrobiología y arqueología[7].

Este proyecto en colaboración consiste en un estudio sobre predicción del origen y profundidad de variscita usando los espectros Raman obtenidos de las muestras. De esta colaboración surgieron varias técnicas y algoritmos que se encuentran en proceso de ser publicados en artículos de investigación. Sin embargo estos resultados no son fácilmente accesibles a aquellas personas sin unos altos conocimientos técnicos en informática.

[1] http://admirable-ubu.es/
[2] http://www.cenieh.es/

Este proyecto busca integrar parte de esos resultados de investigación existentes en una aplicación web, para facilitar el uso de las técnicas y algoritmos desarrollados a los científicos interesados en espectroscopia Raman, de forma que con un par de clicks sean capaces de avanzar en gran medida en su investigación.

Las acciones que este proyecto busca ofrecer son la de visualización, toma de medidas, procesamiento para eliminar ruido, fallos de calibración, fluorescencia, etc y gestión de experimentos de minería de datos.

Aunque inicial y actualmente el proyecto está enfocado en el caso de la geología, con unas clases prefijados, el proyecto podría evolucionar en siguientes versiones para que las clases sean personalizadas y poder usar la aplicación en más ámbitos de la espectroscopia Raman aparte de la geología.

La aplicación web se encuentra disponible en https://spectra-viewer.nanoapp.io/. Se proporciona un cuenta con ejemplos cargados para su uso, el usuario es "tfg.visor.ejemplos@gmail.com" y la contraseña "tfg_visor_ejemplos".

1.1. Materiales entregados

Junto con la memoria se entregan los siguientes materiales:

- Aplicación web "Visor de espectos"
- *Script* de ejecución
- *Script de ejecución de pruebas*
- Ficheros de configuración
- Anexos/documentación técnica

En Internet se encuentran alojados los siguientes recursos:

- Dirección de la aplicación[3]
- Repositorio de la aplicación[4]

[3] https://spectra-viewer.nanoapp.io/
[4] https://github.com/IvanBeke/TFG-Visor-de-espectros

Objetivos del proyecto

Este apartado explica de forma precisa y concisa cuales son los objetivos que se persiguen con la realización del proyecto. Se puede distinguir entre los objetivos marcados por los requisitos del software a construir y los objetivos de carácter técnico que plantea a la hora de llevar a la práctica el proyecto, además de objetivos planteados a nivel personal.

Objetivos principales

- Ofrecer control de usuarios.
- Permitir a los usuarios subir *datasets* y espectros.
- Que los *datasets* y espectros se puedan visualizar.
- Que sobre la visualización se pueda aplicar operaciones de procesamiento.
- Poder entrenar modelos de aprendizaje automático con los *datasets* subidos.
- Poder usar los modelos mencionados anteriormente para predecir nuevos ejemplos subidos.
- Que la aplicación final sea útil para la investigadora.

Objetivos técnicos

- Que la aplicación esté desplegada durante todo el desarrollo.
- Que la aplicación sea fácil de mantener.
- Utilizar git como sistema de control de versiones junto con GitHub como repositorio remoto.
- Utilizar una metodología ágil, Scrum, para el desarrollo.
- Utilizar un sistema kanban para la gestión de tareas.

- Utilizar un sistema de revisión automática de código para asegurar su calidad.

Objetivos personales

- Ampliar los conocimientos sobre desarrollo web a partir de los obtenidos durante el grado.
- Ampliar y profundizar conocimientos sobre Python, especialmente en desarrollo web y aprendizaje automático.
- Aprender a usar técnicas de aprendizaje automático en un entorno de investigación real.
- Desarrollar el proyecto de la forma más profesional posible.

Conceptos teóricos

3.1. Espectroscopia Raman

La espectroscopia Raman hace uso del fenómeno conocido como dispersión inelástica de fotones para obtener gráficas que definen la estructura y composición de un material o elemento.

Este fenómeno se refiere a como los fotones rebotan, o mejor dicho, son absorbidos y vueltos a emitir. Los fotones se pueden dispersar de forma elástica (Rayleigh) o inelástica (Raman).

En la primera forma los fotones absorbidos son emitidos de vuelta igual que fueron absorbidos, la gran mayoría de ellos, pero una pequeña cantidad se emiten cambiados, con una pequeña disminución o aumento de sus energía (ver Figura 3.1).

Este cambio de energía varía según el material o elemento contra el que impacten los fotones, revelando ahí la estructura o composición y abriendo un amplio abanico de aplicaciones para esta técnica[9, 23]. Con la dispersión Raman capturada se obtienen los espectros Raman.

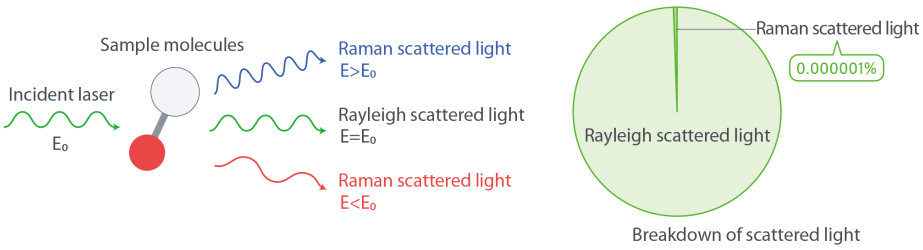


Figura 3.1: Representración de la dispersión de fotones[9]

Con lo explicado anteriormente se puede ver que esta técnica tiene varias ventajas, entre las que destacan[9]:

- Análisis sin contacto y no destructivo.
- No se suele necesitar preparar la muestra
- Sirve para materia orgánica e inorgánica.
- Se puede usar para elementos en cualquier estado de la materia
- Para conseguir un espectro la exposición de la muestra al láser está entre 10ms y 1s

3.2. Visualización de datos

Técnicas usadas con el propósito de presentar datos o información mediante gráficos, como puntos, líneas o barras. Está considerado uno de los pasos dentro del análisis de datos o ciencia de datos[22].

Su objetivo principal es el de comunicas información de forma clara y eficiente, sin significar esto que para que un gráfico sea funcional tenga que parecer aburrido ni que tenga que ser extremadamente sofisticado para resultar agradable[20].

3.3. Minería de datos

La minería de datos se define como la aplicación de técnicas de inteligencia artificial sobre grandes cantidades de datos, con el objetivo de descubrir tendencias, patrones o relaciones ocultas.

Estos descubrimientos suelen ser usados para describir, resumir o clasificar en grupos un conjunto de datos, además de para predecir en que grupo del conjunto encajarían nuevos ejemplos de los datos.

Las fases del proceso de minería de datos se pueden dividir en las siguientes:

1. **Selección**: a partir del conjunto de datos original seleccionar los ejemplos con los que se va a trabajar.
2. **Preprocesamiento**: aplicar operaciones sobre los datos para eliminar ruido o medidas erróneas.
3. **Transformación**: transformar los datos preprocesados a un formato sobre el que poder aplicar las técnicas de minería de datos.

4. **Minería de datos**: aplicar algoritmos sobre los datos capaces de extraer patrones ocultos en ellos.

5. **Evaluación**: comprobar como de bien funcionan los patrones descubiertos en datos nuevos.

Este proyecto se centra en la parte de preprocesamiento, minería de datos y evaluación.

Preprocesamiento

En esta sección se definen las operaciones de preprocesamiento ofrecidas en la aplicación. Se van a explicar tomando como ejemplo uno de los espectros usados por la aplicación.

Recorte (Crop)

La operación de recorte devuelve los valores del espectro contenidos entre un límite inferior y un límite superior (ver figura 3.2).

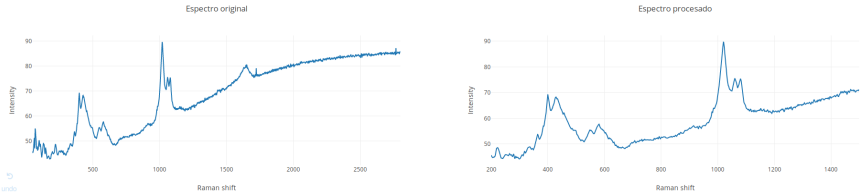


Figura 3.2: Recorte entre valores 200 y 1500

Corrección de línea base (Baseline correction)

La línea base de un gráfico se puede definir como una línea imaginaria sobre la que se apoyan los datos. Esta línea da lugar a mediciones incorrectas por lo que es conveniente eliminarla de los gráficos. Como se ve en la figura 3.3, antes de aplicar la operación, los datos van ascendiendo a medida que aumenta el eje X, después de la operación los datos están correctamente nivelados.

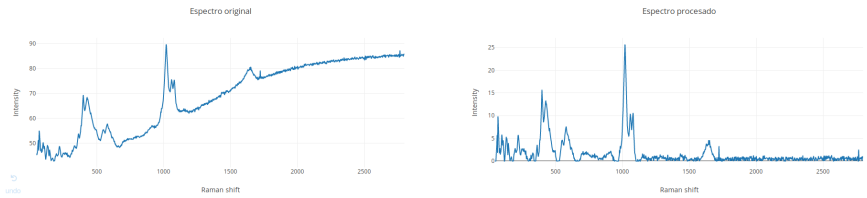


Figura 3.3: Corrección de la línea base

Normalización (Normalization)

La normalización consiste en el proceso de transformar los valores de los espectros de tal forma que todos los espectros se midan por la misma escala para poder ser comparados. Como se ve en la figura 3.4 la escala ha cambiado al rango (0, 1).

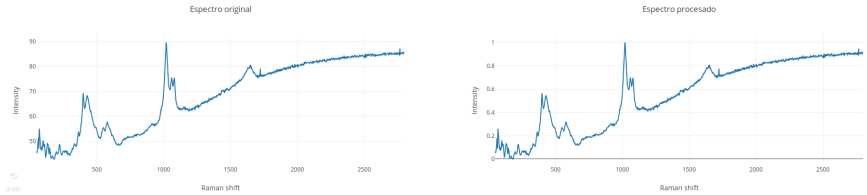


Figura 3.4: Normalización de los datos

Compresión (Squash)

Esta operación consiste en la construcción de datos similares a los originales pero de menor tamaño, sin embargo estos datos nuevos deben producir un resultado casi igual al original al ser analizados. Como se ve en la figura 3.5, el gráfico representado parece el mismo pero la escala es menor, indicando la transformación de los datos.

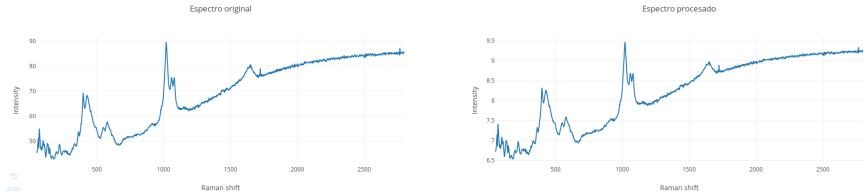


Figura 3.5: Compresión de los datos

Suavizado (Smooth)

La operación de suavizado está enfocada a eliminar el ruido del espectro. Este ruido provoca picos en el gráfico que impiden analizarlo bien ya que pueden modificar la altura de picos que interesan realmente. Esta operación necesita de un parámetro que indica cuanto se van a reducir los picos (ver figura 3.6).

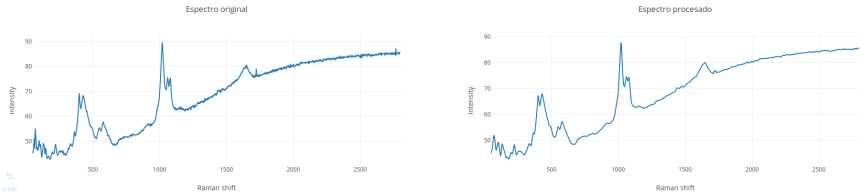


Figura 3.6: Suavizado con ventana de tamaño 25

3.4. Base de datos NoSQL

Las bases de datos no relacionales, o NoSQL (tradicionalmente "non SQL", actualmente "Not Only SQL")[21] son sistemas de almacenamiento de datos que no requieren de una estructura definida y/o fija para su funcionamiento.

Se diferencian principalmente de de los sistemas de bases de datos relacionales en que no necesitan tener definido un esquema al que se ajusten los datos, si no que estos pueden tener la estructura que necesiten y no necesariamente la misma a otros datos almacenados.

Los principales tipos en los que se dividen este tipo de sistemas son:

- Bases de datos documentales
- Bases de datos clave/valor
- Bases de datos en grafo
- Bases de datos orientadas a objetos

Técnicas y herramientas

4.1. Técnicas

Scrum

Scrum es una metodología

Kanban

Despliegue contínuo

4.2. Control de versiones

Sistema de control de versiones

Como sistema de control de versiones

Repositorio remoto

Interfaz gráfico

4.3. Herramientas

Documentación

Calidad del código

Monitorización de dependencias

Entorno de desarrollo

4.4. Almacenamiento

Sistema de almacenamiento

Explorador

4.5. Librerías

4.6. Librerías de representación

Dash

Librería en Python que permite crear sitios webs completos para representación de datos. Para ello hace uso de diversas tecnologías, *Flask* para el servidor web, *Plotly* para la representación y *React* para los componentes y actualización.

Pros

- Gráficos interactivos
- Fácil actualización del gráfico en la web mediante `@app.callback`
- Integración de elementos HTML para la actualización del gráfico
- Uso de la librería *cufflinks* para unir generar una figura directamente de un *DataFrame*

- Al ser de los creadores de *Plotly* y usarlo internamente da la posibilidad de usar sus componentes
- Al usar *Flask* como servidor tiene acceso a todas sus ventajas

Contras

- El código HTML hay que escribirlo desde el código de Python, esto hace que se complique el mantenimiento
- No se pueden reutilizar las plantillas de *Flask*

Plotly

Plataforma para representación de datos, dispone de varias librerías para diferentes lenguajes de programación. Representación online y offline.

Pros

- Gráficos interactivos
- Posibilidad de uso con *Flask* y *Jupyter*

Contras

- Para representar en la web hay que hacer uso de dos versiones de la librería, para Python y para JavaScript
- La representación online guarda los gráficos generados en una cuenta asociada de la plataforma
- La representación offline devuelve el gráfico en Python, pero para representarlo es necesario convertirlo a JSON, enviarlo a la web y que la parte de JS lo represente
- La actualización es necesaria hacerla desde el cliente con JS, donde no se dispone de los datos ni de las utilidades de minería de datos

4.7. Infraestructura

Jupyter Notebook

Aplicación web que permite la edición y ejecución de código, Python en este caso, en el navegador, donde también se muestran el resultado de la ejecución. Dispone de *widgets* para interactuar con el programa. Se instala localmente.

Pros

- Fácil subir archivos al servidor en el menú principal
- Al no tener que hacer una interfaz web permite centrarse en la programación del código de minería de datos
- Los gráficos generados con *Plotly* se representan directamente en el notebook
- Posibilidad de usar `https://mybinder.org/` para el despliegue
- Actualización del gráfico por medio de los *widgets* e `interact`

Contras

- Menos usable e intuitivo
- Al estar el código expuesto el cliente podría alterarlo sin querer
- Solo se puede un usuario en servidor público

Flask

Microframework para aplicaciones web en Python. Aunque por si solo *Flask* no sea muy completo, dispone de una gran cantidad de extensiones oficiales y de la comunidad para suplir todas las características de un framework web completo.

Pros

- Maneja bien la subida de ficheros

- Al ser web hay más control sobre lo que puede hacer el usuario y sobre lo que se le presenta, con la finalidad de hacer más usable la aplicación
- Reutilización de código HTML mediante plantillas y macros

Contras

- Mucho más trabajo al tener que diseñar y programar la interfaz web

4.8. Despliegue

https://www.youtube.com/watch?v=vGphzPLemZE
https://gumroad.com/l/python-deployments
https://www.fullstackpython.com/platform-as-a-service.html
https://www.fullstackpython.com/servers.html

Heroku

Plataforma como servicio, la forma más fácil de despliegue. Tan escalable como fondo tenga la cartera. https://www.heroku.com/

El almacenamiento no es permanente, hay que usar servicios de terceros y conectarlos mediante plugin.

Ngrok

Túnel seguro desde Internet hasta un servidor local en tu máquina. Dirección aleatoria cada vez que se enciende. https://ngrok.com/

El almacenamiento es permanente porque es el almacenamiento de la máquina.

Digital Ocean

Solo de pago pero de momento está disponible por el pack educacional de GitHub. Tan escalable como fondo tenga la cartera. VPS.
https://www.digitalocean.com/
https://pythonprogramming.net/basic-flask-website-tutorial/

Dispone del almacenamiento que ofrece la máquina virtual, es permanente. En caso de que ese espacio sea insuficiente se puede agregar más pagando.

Google App Engine, Google Cloud Platform

Despliegue de Google como plataforma como servicio o VPS. Periodo de prueba gratis y luego tan escalable como fondo tenga la cartera. Funciona con Python 2.7
https://cloud.google.com/appengine/docs/standard/python/getting-started/python-standard-env
https://cloud.google.com/appengine/docs/standard/python/tools/uploadinganapp
https://cloud.google.com/python/getting-started/hello-world
https://cloud.google.com/appengine/docs/flexible/python/quickstart

Ofrece formas de almacenamiento de Google como Cloud Storage para ficheros, caso que nos interesa, hay que pagar por ellas.

Open Shitf

Plataforma como servicio. Plan básico gratis y plan profesional de pago, tan escalable como fondo tenga la cartera.
https://www.openshift.com/
https://blog.openshift.com/beginners-guide-to-writing-flask-apps-on-openshif
https://blog.openshift.com/how-to-install-and-configure-a-python-flask-dev-e

No ofrece almacenamiento por defecto, lo ofrece por medio de lo que llaman "PersistentVolume", ofrecen una API para comunicarse con ello.

PythonAnywhere

Plataforma como servicio especializada en Python. Varios planes, a mejor plan más caro. El plan más básico es gratis.
https://www.pythonanywhere.com/
https://www.youtube.com/watch?v=M-QRwEEZ9-8
Ofrecen almacenamiento de serie pero muy limitado y de pago.

AWS Elastic Beanstalk, AWS CodeStar

Solución en la nube de Amazon. VPS. Tan escalable como fondo tenga la cartera.

https://aws.amazon.com/es/elasticbeanstalk/
https://aws.amazon.com/es/codestar/

Al igual que en el caso de Google, ofrece almacenamiento persistente con sus servicios, Amazon S3, los cuales hay que pagar.

AWS Lambda, Zappa

Zappa es un capa por encima de AWS Lambda para desplegar en modo *serverless*. AWS Lambda se ocupa del escalado y Zappa del despliegue.
https://github.com/Miserlou/Zappa

Solo almacenamiento temporal.

Docker

Despliegue en contenedores.
https://www.docker.com/

De serie no tiene almacenamiento persistente pero es capaz de ofrecerlo mediante "storage drivers". Requiere bastante configuración.

Azure

Solución en la nube de Microsoft. VPS. Tan escalable como fondo tenga la cartera.
https://azure.microsoft.com/es-es/
https://docs.microsoft.com/en-us/azure/app-service/app-service-web-get-started-pyth

Sí ofrece almacenamiento persistente.

Nanobox

Solución interesante, combina los contenedores de Docker con despliegue en la nube y lo automatiza. De momento compatibilidad con Digital Ocean, Amazon y Linode, Google, Joyent y Azure en camino. Plan básico gratis, el resto de precios son flexibles. También en local.
https://nanobox.io/

https://github.com/nanobox-io/nanobox

Ofrece almacenamiento persistente, hay que configurar las rutas que van a ser persistentes en el fichero de configuración. Cada despliegue el almacenamiento que se haya usado so borra. Depende del almacenamiento que esté disponible en el servicio escogido para almacenar.

Aspectos relevantes del desarrollo del proyecto

En este apartado se van a recoger los aspectos más importantes del desarrollo del proyecto. Desde las decisiones que se tomaron y sus implicaciones, hasta los numerosos problemas a los que hubo que enfrentarse y cómo se solucionaron.

5.1. Elección del proyecto

A finales del curso pasado se organizó una charla en la que los profesores iban a presentar las optativas que daban clase de forma que los alumnos tuviéramos más fácil elegir asignaturas. En la presentación de la asignatura "Minería de Datos" despertó interés por el tema y se preguntó a José Francisco, por haber realizado la presentación, sobre TFGs relacionados con el tema.

De los trabajos disponibles este llamó la atención por estar relacionados con geología y con desarrollo web, además de poder aplicar técnicas minería de datos en un entorno real de investigación.

5.2. Formación

Para poder realizar el proyecto se necesitaban unos conocimientos no adquiridos sobre desarrollo web, tanto de la parte de servidor en Flask como la parte del cliente en HTML, CSS y JavaScript, aunque en menor medida por haberse tocado algo durante el grado. Como se había hablado con los tutores antes de verano sobre el proyecto, se dedicó parte a aprender sobre

ello. Además de para aprendizaje, los recursos se han usado también como material de consulta durante el desarrollo.

Para la parte del servidor se siguieron los libros y tutoriales:

- Flask Web Development[14]
- Explore Flask[19]
- The Flask Mega-Tutorial Legacy (2012)[13]
- The Flask Mega-Tutorial (2017)[15]

Para la parte del cliente se utilizaron principalmente los siguientes materiales:

- MDN Web Docs[18]
- W3Schools Tutorials[8]

A medida que se añadían nuevas herramientas al proyecto, su documentación oficial también ha sido consultada en varias ocasiones, están disponibles en:

- Documentación de Flask[2]
- Documentación de Bootstrap[11]
- Documentación de Nanobox[3]
- Documentación de PyMongo[5]
- Documentación de Dash[1]
- Documentación de Plotly[4]

5.3. Sistema de usuarios

Uno de los primeros problemas que se plantearon fue la forma de ofrecer el sistema de usuarios, para que cada uno pudiera almacenar sus archivos. Las opciones que se presentaban eran implementar uno desde cero con ayuda de las extensiones que ofrece Flask o mediante un sistema de terceros, como puede ser Google.

Implementar el sistema desde cero tenía la ventaja de que los usuarios no tenían que salir de la página para iniciar sesión y que se había aprendido como hacerlo en los tutoriales sobre Flask mencionados anteriormente, mientras que el sistema de terceros no se sabía como hacerlo, pero tenía más ventajas, siendo las principales no tener que mantener una base de datos de usuarios, no depender de un sistema de envío de correo electrónico (dio problemas en proyectos anteriores) y no tener que obligar a los usuarios a crearse otra cuenta al poder usar una existente.

Al final se decidió usar la autenticación con Google, por estar casi garantizado que los usuarios van a tener una cuenta existente, la documentación sobre este aspecto es abundante y era fácil de implementar. Además al iniciar sesión con este servicio nos permite usar la API de Google para obtener los datos del usuario necesarios.

5.4. Cohesión entre Flask y Dash

Al contar con la presencia de dos *frameworks* de desarrollo web hacer que funcionen juntos ha sido un reto, sobre todo teniendo en cuenta que Dash se ejecuta Flask pero no permite reutilizar sus componentes, favoreciendo la aparición de código repetido y disminuyendo la mantenibilidad.

El mayor problema que se ha tenido ha sido la diferencia en como escribir el código HTML para la interfaz en el cliente. Mientras que en Flask se escribe en ficheros HTML llamados *templates*, Dash apuesta por escribir todo el código desde Python. Esto favorece que surjan defectos de código, sobre todo de código duplicado.

Se investigó bastante sobre cómo poder usar los *templates* de Flask para definir la interfaz de Dash pero no se pudo encontrar forma de poder hacerlo. Al final no quedó otra forma de hacerlo, pero dentro de lo malo es un pequeño precio a pagar por el resto de ventajas que Dash ofrece.

5.5. Subida de ficheros

Una de las primeras características que se implementó fue la subida de ficheros al servidor, al tener ya el sistema de usuarios se podían organizar los datos sin problema para cada cliente. El formato requerido para subir los datos pasó por varios cambios motivados por la forma en la que Susana nos enviaba los datos que probar la aplicación. En secciones posteriores se

explica como estos cambios afectaron también al almacenamiento de los datos y la interfaz del visor.

Organización en directorios

Al principio los datos se encontraban organizados en directorios cuyo nombre proporcionaba toda la información sobre los ejemplos que contenían. Por lo que el formato que se pedía era simple, un archivo comprimido en formato ".zip" que contuviera estos directorios.

Estructura según hoja de metadatos

Sobre mediados de Abril los nuevos datos recibidos empezaron a estar organizados según una hoja de Excel que relacionaba los nombres de directorios y los ejemplos que contenían con sus metadatos, en vez de estar contenidos en el nombre.

Esta hoja contenía una columna llamada "Id" con el nombre de un directorio y los ejemplos de ese tipo, las siguientes columnas contenían datos sobre los ejemplos contenidos en esa fila, como la mina de la que han sido extraídas las muestras o la profundidad.

Esta hoja se adaptó para aplicación manteniendo la columna "Id" y añadiendo fijas una columna para la mina y otras dos para la profundidad, una nominal y otra numérica. El formato de subida pasó a ser en un mismo fichero ".zip" los directorios que contienen los ejemplos y la hoja con los metadatos previamente cumplimentada.

5.6. Interfaz del visor

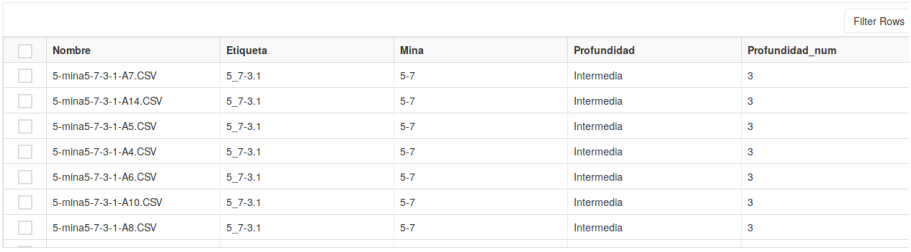
Esta parte del proyecto está muy relacionado con lo explicado en la sección 5.5, el usuario del visor espera que se le presenten los datos de una forma similar a como los ha subido.

Cuando los datos se subían como un archivo comprimido con carpetas cuyo nombre contiene la información se presentaban al usuario dos menús desplegables. En la carga solo uno de ellos contenía valores, los nombres de los directorios. El segundo desplegable actualizaba sus valores con los nombres de los ficheros contenidos en la carpeta seleccionada, al seleccionar el fichero se cargaba en el visor.

Con el cambio de la hoja con metadatos, el tutor José Francisco propuso que al usuario se le mostrase una tabla parecida al hoja que había rellenado

antes de subir los datos, con la diferencia de que se mostrasen todos los ficheros como filas con sus metadatos, asemejándose a la estructura final de almacenamiento de datos.

Mostrar los datos al usuario en forma de tabla trajo las ventajas de poder representar varios espectros a la vez, filtrar y ordenar según las columnas (ver Figura 5.7).


Filter Rows

Nombre	Etiqueta	Mina	Profundidad	Profundidad_num
5-mina5-7-3-1-A7.CSV	5_7-3.1	5-7	Intermedia	3
5-mina5-7-3-1-A14.CSV	5_7-3.1	5-7	Intermedia	3
5-mina5-7-3-1-A5.CSV	5_7-3.1	5-7	Intermedia	3
5-mina5-7-3-1-A4.CSV	5_7-3.1	5-7	Intermedia	3
5-mina5-7-3-1-A6.CSV	5_7-3.1	5-7	Intermedia	3
5-mina5-7-3-1-A10.CSV	5_7-3.1	5-7	Intermedia	3
5-mina5-7-3-1-A8.CSV	5_7-3.1	5-7	Intermedia	3

Figura 5.7: Tabla de espectros

5.7. Almacenamiento de datos

El principal motivo de implementar un sistema de usuarios fue que cada uno pudiera tener sus datos almacenados para no tener que subirlos cada vez que se quiera trabajar con ellos.

Inicialmente

Al principio se almacenaban directamente en un directorio nombrado como el id del usuario y lo único que se hacía era descomprimir el archivo comprimido en ese directorio. Esta forma de almacenar los datos conlleva varios problemas que motivaron el cambio en la forma de almacenamiento a la actual. Esta forma de almacenamiento se corresponde al formato explicado en la sección 5.5 (Organización en directorios).

El problema más evidente, y que más notarían los usuarios, es que cada vez que se quisiera ver un espectro hay que cargarlo desde el disco, disminuyendo el rendimiento general de la aplicación.

Otro gran problema de esta forma de almacenar era como manejaba el servidor los ficheros guardados, este tema se explicará más en detalle en la sección 5.9.

El último problema y el más problemático desde el punto de vista de programación es que para cada operación que se quisiera realizar con los

datos se necesitaban escribir funciones auxiliares complejas para buscar y manipular el árbol de directorios, las cuáles había que modificar con cada cambio en la estructura, dificultando el mantenimiento de la aplicación.

Se planteó cambiar la forma de almacenar los datos en el servidor, cambio que terminó por realizarse después de valorar ventajas y desventajas en este momento del desarrollo.

Migración a MongoDB

Por sugerencia del tutor se investigó la posibilidad de usar MongoDB para almacenar los datos. Se vio que podría funcionar realmente bien al poder insertar directamente los ficheros cargados en la base de datos, recuperarlos y borrarlos fácilmente. Como consecuencia directa del cambio los métodos auxiliares con los que se interactuaba con los datos disminuyeron considerablemente en tamaño y complejidad.

En primera instancia se optó por guardar los conjuntos de datos como un documento en el que estaban contenidos varios *DataFrames* (cada uno representando un espectro) agrupados según la carpeta en la que estuvieran localizados los espectros. Cada *DataFrame* contenía dos columnas, una con los valores del eje X y otra con los valores del eje Y.

Pero seguido de adoptar esta estructura y tener la aplicación adaptada para ello llegó el cambio en como recibíamos los datos, provocando otra modificación en la estructura. Esta vez se decidió agrupar todo el conjunto de datos subido en un solo *DataFrame*, en el que cada fila se corresponde con un espectro, las columnas representan el eje X y los valores de cada fila en esas columnas representan el valor del eje Y. Cada fila contiene adicionalmente columnas que indican el nombre del espectro y sus metadatos asociados (ver Figura 5.8). En la sección 5.8 se explican más en profundidad este cambio.

...	2796	2797	2798	2799	2800	Label	Name	Mina	Profundidad	Profundidad_num
...	66.037154	65.891367	65.902817	66.131317	66.392963	5_7-3.1	5-mina5-7-3-1-A10.CSV	5-7	Intermedia	3
...	76.284811	76.048776	76.057184	76.478205	76.818416	5_7-3.1	5-mina5-7-3-1-A11.CSV	5-7	Intermedia	3
...	63.994108	64.014814	63.907530	63.779326	63.775762	5_7-3.1	5-mina5-7-3-1-A12.CSV	5-7	Intermedia	3
...	85.739489	85.754369	85.744979	85.796215	85.799102	5_7-3.1	5-mina5-7-3-1-A14.CSV	5-7	Intermedia	3
...	140.914040	141.203355	141.390541	141.054794	140.673620	5_7-3.1	5-mina5-7-3-1-A2.CSV	5-7	Intermedia	3

Figura 5.8: Estructura definitiva

Guardar los datos de esta forma facilita aplicar los métodos de minería de datos al estar pensado para aplicarse en lote.

5.8. Integración de los algoritmos existentes

Como se ha comentado en la introducción (página 1), este proyecto surge de una colaboración. De aquí se obtuvieron bastantes algoritmos, gran parte de ellos dedicados al procesamiento de los espectros, que se añadieron sobre la librería "superman" (6.3).

Una de las partes más importantes del proyecto era conseguir integrar estos algoritmos en la aplicación web para poder ejecutarlos sobre los espectros visualizados.

Cuando llegó el momento de integrarlos algoritmos en la aplicación había dos opciones para hacerlo funcionar, modificar la aplicación para adaptarse a los algoritmos o modificar los algoritmos para adaptarse a la aplicación.

En primera instancia se intento modificar los algoritmos pero se vio que era un paquete con demasiadas dependencias dentro del propio paquete, por lo que cada cambio producía gran cantidad de errores de funcionamiento, que al solucionar producían todavía más errores dentro del paquete. Se optó por revertir estos cambios y modificar la aplicación.

La modificación principal fue la modificación final en como guardar los datos comentada anteriormente (figura 5.8). Con esa modificación el funcionamiento de la aplicación volvió a ser el correcto.

5.9. Despliegue

La idea de realizar el proyecto como una aplicación web tiene relación más que directa con que pueda estar accesible en pocos clicks. Para ello se necesita que esté desplegada y accesible en Internet. En esta sección de describe las etapas por las que pasó el despliegue, los problemas que surgieron y como se solucionaron.

En las primeras reuniones del proyecto se comentó con los tutores que un gran problema de los proyectos anteriores desarrollados en web se centraban en el despliegue al final del proyecto, haciendo que alguna vez no pudieran llegarse a desplegar, por eso se planteó la idea de empezar a desplegar desde el inicio del proyecto.

Servidor

La plataforma escogida fue Heroku[5] por su simplicidad y un plan gratuito que cubre las necesidades del proyecto. Aunque en primera instancia parecía que esta plataforma funcionaba bien para nuestras necesidades se vio que no contaba con almacenamiento persistencia, convirtiendo su uso en inviable.

Las opciones que se plantearon fueron buscar una forma de añadir este almacenamiento y cambiar de servidor por completo. Para añadirlo en Heroku había que depender de *plugins* de terceros para enlazar servicios de almacenamiento de terceros teniendo que modificar la aplicación para hacerlo funcionar, además de ser servicios de pago.

Al final se escogió por cambiar a un proveedor *cloud* de pago que ofreciese máquinas virtuales privadas, las opciones manejadas fueron DigitalOcean[6], Amazon Web Services[7] y Google Cloud[8]. Se escogió la primera opción ya que gracias al Student Developer Pack[9] se disponía de un cupón de 50$ en esta plataforma.

Herramientas para el despliegue

La plataforma Heroku posee sus propias herramientas para el despliegue, facilitando en gran cantidad este proceso. Esta plataforma un repositorio *git* remoto para almacenar la aplicación por lo que al contar ya con este sistema de control de versiones en el proyecto no hubo que modificar casi nada para poder desplegar, pero por lo problemas comentados anteriormente se tuvo que abandonar.

Para complementar a DigitalOcean y facilitar la tarea del despliegue se ha usado el servicio Nanobox[10]. Al principio de usar esta plataforma se vio que los datos almacenados se eliminaban en cada despliegue, para ello hubo que añadir un segundo contenedor que se ocupara del almacenamiento, lo bueno es que este contenedor se asocia a un directorio del servidor, por lo que la aplicación no se tuvo que modificar. Más adelante se añadió otro contenedor para gestionar la base de datos MongoDB.

[5] https://www.heroku.com/
[6] https://www.digitalocean.com/
[7] https://aws.amazon.com/es/
[8] https://cloud.google.com/
[9] https://education.github.com/pack
[10] https://nanobox.io/

5.10. Creación de clasificadores

Dentro del objetivo de aplicación de técnicas de minería de datos había que decidir si dejar a los usuarios la opción de crear clasificadores personalizados o simplemente entrenarlos a partir de *datasets* subidos. Se decidió probar la primera opción, teniendo la segunda como respaldo en caso de no funcionar.

Obtención de parámetros

Para conseguir esta personalización surge el problema de cómo obtener los parámetros del clasificador y presentárselos al usuario. La primera idea que surgió es analizar la documentación oficial de los clasificadores y crear a mano un fichero con los parámetros, esto puede resultar viable si se usan pocos clasificadores pero dificulta mucho la adición de nuevos.

Se descartó a favor de buscar una herramienta que pudiera analizar la documentación *in-code* de los clasificadores. Al ser un formato estructurado y conociendo herramientas en otros lenguajes de análisis de este tipo de documentación seguro que para Python existen herramientas parecidas.

Se encontró la librería numpydoc[11] que, entre otras características, ofrece la funcionalidad que se busca, analiza la documentación y devuelve un diccionario con las secciones, entre ellas los parámetros.

Con esta información se puede generar un formulario automáticamente y hacer que cambie dinámicamente según la opción del usuario.

Procesamiento del formulario

El servidor recibe todos los datos como si fueran cadenas, aunque el control en el formulario sea de tipo numérico, por lo toca convertir las cadenas al tipo correcto con un método de prueba y error, para luego guardarlos y pasárselos al constructor del clasificador. Si no se ha introducido valor no se guarda permitiendo usar el valor por defecto.

Primero si el valor coincide con un *booleano* se convierte, si no se prueba a convertir en entero, si salta excepción se prueba a convertir en numérico con decimales, en caso de que salte excepción, al no haber más tipos primitivos se asume que el valor tiene que ser una cadena y guarda como tal.

[11] https://github.com/numpy/numpydoc

Mantener el modelo entre peticiones

Al ser los clasificadores creados y personalizados desde la web, estos necesariamente se tienen que crear mientras se procesa la petición, con su consecuente destrucción al terminar de procesarla, con lo que la tarea de guardarlos se complica.

Las ideas propuestas para solventar el problema fueron guardar los parámetros usados como una *cookie* para poder entrenar el modelo otra vez antes de guardarlo o serializar el modelo ya entrenado en el servidor temporalmente y cargarlo en caso de querer guardarlo definitivamente.

A pesar de la facilidad de usar *cookies* para solucionar el problema, estos datos guardados son necesarios solo en el caso de que se elija guarda el modelo, por lo que parece algo precipitado guardarlos cuando puede que no se lleguen a usar.

Sin embargo la razón para descartar esta idea es la confianza que se ofrece al usuario. Si se reentrena el modelo, aunque sea con los mismos parámetros, la separación que se hace de los datos en entrenamiento y test puede ser diferente, resultando en un clasificador diferente con estadísticas diferentes.

La opción de serialización permite centralizar todo en el servidor, permitiendo una fácil carga y guardado del modelo entrenado, solucionando el problema de la confianza. Además se evita el envío de datos innecesarios al usuario en cada petición. Por último, el servidor está configurado para vaciar el almacenamiento temporal cada día evitando el problema de almacenar datos que puede que no se usen, porque la decisión de guardar el clasificador se toma justo después de evaluarlo.

Trabajos relacionados

Como se comentó en la introducción (página 1), el uso de técnicas de minería de datos está empezando a crecer su uso en este campo. Primero se comenta un artículo que habla en profundidad sobre este tema para seguido hablar de herramientas existentes para el análisis de espectros Raman.

6.1. Artículos científicos

Machine learning tools for mineral recognition and classification from Raman spectroscopy[10]

En este estudio se prueban a usar técnicas de aprendizaje automático con el objetivo de mejorar la identificación de materiales usando todo el rango del espectro. Se enumeran varias técnicas preprocesamiento y algoritmos de clasificación usados para comprobar con que combinaciones se obtienen mejores resultados. Las técnicas de procesamiento de este proyecto están basadas en las indicadas en este artículo.

6.2. Software

CrystalSleuth

Este software desarrollado dentro del proyecto RRUFF[12], dedicado a recopilar en una base de datos espectros Raman, difracción de rayos X y datos químicos de minerales, permite cargar espectros para su análisis y

[12] http://rruff.info/

manipulación, adicionalmente pueden compararse los espectros cargados con los almacenados para intentar averiguar a qué pertenecen.

6.3. Librerías

Scikit-spectra

Al principio del proyecto se habló mucho con los tutores de probar y usar esta librería como referencia o como base del proyecto. Pero después de analizar su repositorio se vio que llevaba tiempo sin mantenimiento y al instalar y probar los ejemplos que trae incluidos salían errores por todas partes, por lo que se dejó de lado. La principal característica de la librería es la visualización y la construcción de interfaces gráficas mediante *IPython Notebooks* para su ejecución en navegador[16].

Superman

Abreviatura de "SpectrUm PrEpRocessing MAchiNe". Esta librería alojada en GitHub[13] fue usada en el desarrollo del proyecto publicado en el artículo mencionado anteriormente (6.1)[10], siendo el autor del artículo uno de los desarrolladores. Se ha usado como librería base para las opciones de procesamiento, añadiendo funciones según el formato existente.

[13] https://github.com/all-umass/superman

Conclusiones y Líneas de trabajo futuras

En esta sección se exponen las conclusiones obtenidas al terminar el desarrollo del proyecto y se comentan anotaciones que podrían seguirse para avanzar con este trabajo en el futuro.

7.1. Conclusiones

A una semana de la entrega y habiendo terminado el desarrollo del proyecto puedo considerar que se han cumplido los objetivos definidos y el producto resultante es una aplicación que va a facilitar el trabajo de Susana, la geóloga colaboradora, además de una herramienta de minería de datos que puede llegar a ser muy útil para gente que sin muchos conocimientos de minería de datos pueda usar estas técnicas sin problema.

También puedo afirmar que este proyecto ha sido bastante ambicioso respecto a todo lo que se quería que ofreciese, provocando que se centrase demasiado en añadir todas las funcionalidades y descuidando, por desgracia, algunas cuestiones referentes a seguridad, tolerancia a fallos, diseño y pruebas del sistema, cuestiones planteadas como líneas futuras. Aún así estoy muy contento del producto final y de todo lo que ofrece.

Como la etapa final de aprendizaje del grado destacar la cantidad enorme de conocimientos nuevos que se adquieren. En mi caso este conocimiento se centra principalmente en desarrollo web, *front-end* y *back-end*, conocimientos que era de obligatorio cumplimiento su adquisión. Me parece que estos temas deberían explorarse durante el grado por, en mi opinión, ser unos conocimientos básicos que todo graduado en informática debería poseer.

Otros conocimientos adquiridos han sido el manejo de bases de datos NoSQL, en concreto bases documentales, un tema sobre el que no se esperaba aprender pero que seguro es de gran utilidad en el futuro, personal y profesionalmente.

El último punto que quiero comentar, y de los que más ilusión me ha traído durante el desarrollo es ver, desarrollar y aplicar técnicas de minería de datos a ejemplos reales de investigación. Aunque día a día salgan noticias sobre avances en inteligencia artificial parece algo lejano, a veces casi de ciencia ficción. Pero ser capaz de aplicar estas técnicas, aprender cómo funcionan y, sobre todo, poder decir frases como "El proyecto que he desarrollado es capaz de decirte la profundidad a la que se ha extraído un mineral" consigue hacer de esta disciplina algo más cercano.

7.2. Líneas de trabajo futuras

Este trabajo esta pensado para evolucionar en el futuro hacia un proyecto más ambicioso de tal forma que sea posible su uso en otros ámbitos aparte de la geología. A continuación se presenta una lista de tareas a realizar para continuar con su desarrollo:

- Mejorar la seguridad y tolerancia a fallos en las búsquedas a la base de datos.
- Mejorar el diseño general de los modelos.
- Modificar la aplicación de forma que sea capaz de trabajar con atributos definidos por el usuario, en vez de que estos estén fijos.
- Mejorar la creación de clasificadores para poder elegir sobre que atributo crearlos, en vez de todos a la vez.
- Poder aplicar un preprocesamiento de los datos antes de crear el clasificador en vez de usar uno fijo y por defecto.
- Crear unos tests más exhaustivos.
- Mejorar el sistema de usuarios añadiendo opciones como "Eliminar cuenta" o similares.
- Poder poner tareas que requieran de más tiempo en segundo plano para que el usuario pueda seguir usando la aplicación.

- Teniendo en cuenta el punto anterior, añadir un sistema de notificaciones por correo electrónico para avisar al usuario que las tareas que estaban en segundo plano han terminado.

Bibliografía

[1] Dash user guide. https://dash.plot.ly/.

[2] Flask documentation. http://flask.pocoo.org/docs/1.0/.

[3] Nanobox documentation. https://docs.nanobox.io/.

[4] Plotly user guide. https://plot.ly/python/user-guide/.

[5] Pymongo documentation. https://api.mongodb.com/python/current/.

[6] Susana Jorge Villar | CENIEH - Centro Nacional de Investigación sobre la Evolución Humana. http://www.cenieh.es/es/personal/susana-jorge-villar.

[7] Susana Jorge Villar | CV Docente. http://apps.ubu.es/profesorado/cv.php?docente=seju@ubu.es.

[8] W3Schools. https://www.w3schools.com/.

[9] What is Raman Spectroscopy? | Nanophoton. https://www.nanophoton.net/raman/raman-spectroscopy.html.

[10] Carey C., Boucher T., Mahadevan S., Bartholomew P., and Dyar M. D. Machine learning tools formineral recognition and classification from raman spectroscopy. *Journal of Raman Spectroscopy*, 46(10):894–903.

[11] Bootstrap Contibutors. Bootstrap documentation. https://getbootstrap.com/docs/3.3/.

[12] Supratim Dey. How is raman spectroscopy useful to geology? https://www.quora.com/How-is-Raman-spectroscopy-useful-to-geology, 2015. [Online; accessed 25-May-2018].

[13] Miguel Grinberg. The Flask Mega-Tutorial Legacy. https://blog.miguelgrinberg.com/post/the-flask-mega-tutorial-part-i-hello-world-legacy, 2012.

[14] Miguel Grinberg. *Flask Web Development: Developing Web Applications with Python*. O'Reilly Media, 2014.

[15] Miguel Grinberg. The Flask Mega-Tutorial. https://blog.miguelgrinberg.com/post/the-flask-mega-tutorial-part-i-hello-world, 2017.

[16] Adam Hughes, Mark Reeves, and Zhaowen Liu. Scikit-spectra: Explorative spectroscopy in python. *Journal of Open Research Software*, 3, 06 2015.

[17] Priceton Instruments. Raman Spectroscopy Basics. http://web.pdx.edu/~larosaa/Applied_Optics_464-564/Projects_Optics/Raman_Spectrocopy/Raman_Spectroscopy_Basics_PRINCETON-INSTRUMENTS.pdf.

[18] Mozilla. MDN Wen Docs. https://developer.mozilla.org/en-US/.

[19] Robert Picard. Explore flask. https://exploreflask.com/en/latest/, 2014.

[20] Vitaly Friedman. Data Visualization and Infographics. https://www.smashingmagazine.com/2008/01/monday-inspiration-data-visualization-and-infographics/, 2008. [Online; accessed 24-June-2018].

[21] Wikipedia. Nosql — wikipedia, la enciclopedia libre, 2018. [Internet; descargado 24-junio-2018].

[22] Wikipedia contributors. Data visualization — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Data_visualization&oldid=844657354, 2018. [Online; accessed 24-June-2018].

[23] Wikipedia contributors. Raman scattering — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Raman_scattering&oldid=841233599, 2018. [Online; accessed 25-May-2018].

[24] Wikipedia contributors. Raman spectroscopy — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Raman_spectroscopy&oldid=839660612, 2018. [Online; accessed 25-May-2018].