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 — 19 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, 19 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 de figuras

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[16], generalmente en el rango de la luz visible o cercano al infrarrojo o ultravioleta[20]. 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.[11].

En el área de lenguajes y sistemas informáticos existen colaboraciones en curso entre el grupo de investigación ADMIRABLE[1] y 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, 19]. Con la dispersión Raman capturada se obtienen los espectros Raman.

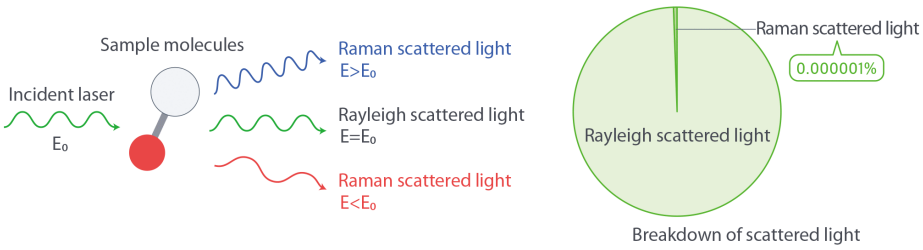

Figura 3.1: Representración de la dispersión de fotones[9]

Herramientas	App AngularJS	API REST	BD	Memoria
HTML5	X			
CSS3	X			
BOOTSTRAP	X			
JavaScript	X			
AngularJS	X			
Bower	X			
PHP		X		
Karma + Jasmine	X			
Slim framework		X		
Idiorm		X		
Composer		X		
JSON	X	X		
PhpStorm	X	X		
MySQL			X	
PhpMyAdmin			X	
Git + BitBucket	X	X	X	X
MikTEX				X
TEXMaker				X
Astah				X
Balsamiq Mockups	X			
VersionOne	X	X	X	X

Tabla 3.1: Herramientas y tecnologías utilizadas en cada parte del proyecto

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

Técnicas y herramientas

4.1. 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

4.2. 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

4.3. 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-openshift/
https://blog.openshift.com/how-to-install-and-configure-a-python-flask-dev-environm

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-start

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[13]
- Explore Flask[18]
- The Flask Mega-Tutorial Legacy (2012)[12]
- The Flask Mega-Tutorial (2017)[14]

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

- MDN Web Docs[17]
- 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:

- Flask[2]
- Bootstrap[10]
- Nanobox[3]
- PyMongo[5]
- Dash[1]
- 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. Despliegue

Heroku

Problemas

Nanobox como solución

5.5. Procesamiento

5.6. Tecnología y frameworks

Decisiones

Compatibilidad

Cohesión

5.7. Almacenamiento de datos

Inicialmente

Problemas

Migración a MongoDB

Primera aproximación

Estructura definitiva

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.

En este apartado se hace un pequeño resumen o mención de otras herramientas o artículos sobre el tema que puedan estar relacionados con el tema tratado en el proyecto.

6.1. 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[15].

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

[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/3.6.1/.

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

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

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

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

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

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

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

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

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

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

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