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 mayo 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. nombre tutor, profesor del departamento de nombre departamento, área de nombre área.

Expone:

Que el alumno D. Iván Iglesias Cuesta, con DNI dni, ha realizado el Trabajo final de Grado en Ingeniería Informática titulado título de TFG.

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 mayo de 2018

Vº. Bº. del Tutor:	Vº. Bº. del co-tutor:
D. nombre tutor	D. nombre co-tutor

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, procedencia 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, variscitas, 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

Este proyecto surge de una colaboración previa entre una investigadora en geología y los tutores, Dr. José Francisco Díez Pastor y Dr. César Ignacio García Osorio. La colaboración consistía en un proyecto sobre análisis de un mineral llamado variscita mediante espectroscopia Raman.

1.1. Estructura de la memoria

1.2. Materiales entregados

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.

En esta sección se listan los objetivos que persigue la realización del proyecto, tanto

Objetivos principales

- Desarrollar una aplicación que proporcione las opciones de procesamiento más comunes.
- Que la aplicación final sea útil para la investigadora.
- La aplicación tiene que ser usable e intuitiva.
- Que la herramienta se consiga desplegar.
- Ofrecer control de usuarios para que cada uno pueda almacenar sus archivos.
- Ofrecer un sistema de aprendizaje automático para ayudar en la clasificación de futuras muestras de espectros.

Objetivos técnicos

- Que la aplicación sea fácil de mantener.
- Utilizar git como sistema de control de versiones junto con GitHub para el 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.
- Utilizar un sistema de integración continua.

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

En aquellos proyectos que necesiten para su comprensión y desarrollo de unos conceptos teóricos de una determinada materia o de un determinado dominio de conocimiento, debe existir un apartado que sintetice dichos conceptos.

Algunos conceptos teóricos de LaTeX[1].

3.1. Secciones

Las secciones se incluyen con el comando section.

Subsecciones

Además de secciones tenemos subsecciones.

Subsubsecciones

Y subsecciones.

3.2. Referencias

Las referencias se incluyen en el texto usando cite [**?**]. Para citar webs, artículos o libros [**?**].

[1] Créditos a los proyectos de Álvaro López Cantero: Configurador de Presupuestos y Roberto Izquierdo Amo: PLQuiz

3.3. Imágenes

Se pueden incluir imágenes con los comandos standard de LaTeX, pero esta plantilla dispone de comandos propios como por ejemplo el siguiente:

Figura 3.1: Autómata para una expresión vacía

3.4. Listas de items

Existen tres posibilidades:

- primer item.
- segundo item.

1. primer item.
2. segundo item.

Primer item más información sobre el primer item.

Segundo item más información sobre el segundo item.

-

3.5. Tablas

Igualmente se pueden usar los comandos específicos de LaTeXo bien usar alguno de los comandos de la plantilla.

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

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

5.1. Introducción

5.2. Tecnología y frameworks

Decisiones

Compatibilidad

Cohesión

5.3. Almacenamiento de datos

Inicialmente

Problemas

Migración a MongoDB

Primera aproximación

Estructura definitiva

5.4. Despliegue

Heroku

Problemas

Nanobox como solución

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