Clasificador de música.

Ramón Manuel Redondo Rodríguez

Sistemas de Información de Gestión y Business Intelligence

Índice

Introducción

El objetivo de este informe consiste en explicar y documentar todo el proceso de creación de una aplicación de software usando TensorFlow, en este caso un clasificador de música.

La aplicación consiste en un software basado en la librería TensorFlow de Google que, a partir de unos parámetros determinados, reconozca y clasifique canciones en dos categorías: las que le pueden gustar a un usuario y las que no.

Esta aplicación esta escrita en Python, pero se nutre de librerías externas como TensorFlow, mencionada anteriormente, o spotipy, una librería para el manejo de la API de Spotify. Spotipy se va a utilizar para la extracción de datos, es decir, la extracción de las canciones y las variables que posteriormente se analizarán. TensorFlow se encargará del aprendizaje.

Cronograma y objetivos

El proceso de desarrollo abarca un tiempo parejo al de la asignatura Sistemas de Información de Gestión y Business Intelligence a la que va dirigido el proyecto.

Todo este desarrollo se divide en una serie de tareas, que se comprenden desde el inicio de la asignatura en septiembre del 2017, hasta la presentación y fin en enero de 2018. Estas tareas son las siguientes:

- Estudio de TensorFlow
- Estudio de Python
- Instalación de TensorFlow y Python
- Estudio del material de aprendizaje (videos y libros)
- Elección de un tema para la aplicación
- Estudio de la API de Twitter
- Desarrollo de código

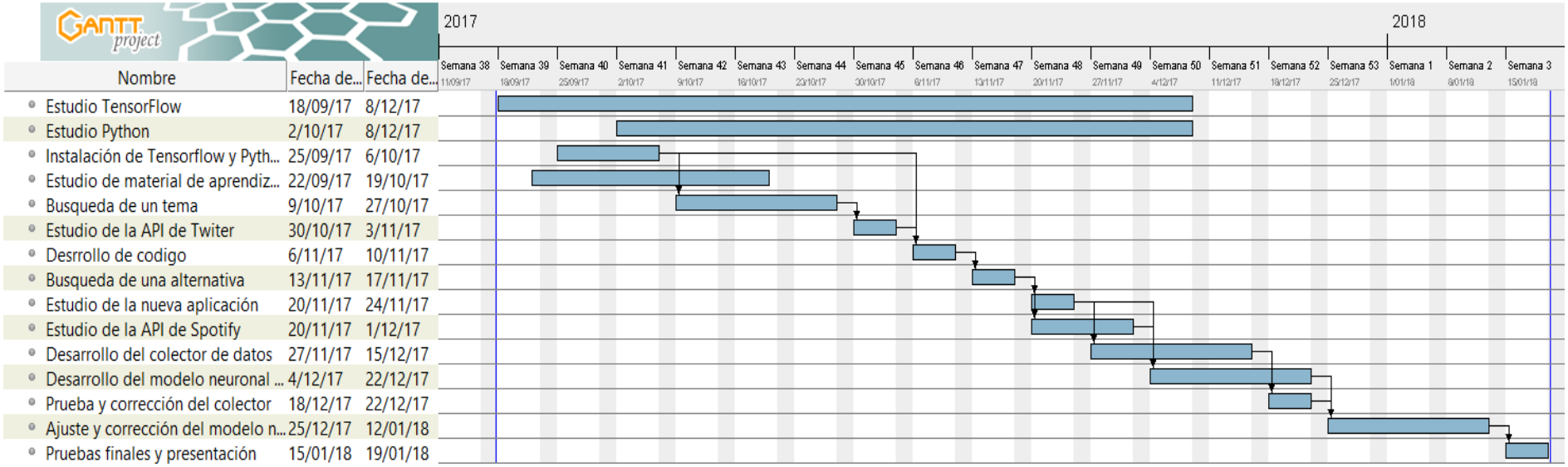
- Búsqueda de un tema alternativo
- Estudio del nuevo tema
- Estudio de la API de Spotify
- Desarrollo del colector de datos
- Desarrollo del modelo neuronal (TensorFlow)
- Prueba y correcciones del colector
- Ajuste y correcciones del modelo neuronal
- Pruebas finales y corrección

Como se puede ver en el diagrama de Gantt de la siguiente página, todo el proceso de desarrollo tiene unas fases de aprendizaje que abarcan gran parte del tiempo del proyecto y que se solapan con otras tareas. Esto es debido a que, tanto en este desarrollo como en cualquiera, el aprendizaje es constante, aunque este más focalizado en las fases tempranas, se da durante todo el proceso.

Además de estas fases de aprendizaje, también se pueden ver tres fases que no suponen un avance real puesto que en el proceso de desarrollo hubo un cambio de objetivo debido a la dificultad y a la escasez de librerías de preprocesado de texto en español para llevar a cabo la primera opción: un análisis de sentimiento en Twitter.

El objetivo es finalizar la aplicación en la segunda semana de enero para tener algo de tiempo de margen por si ocurre algún problema o hay algún retraso en alguna de las tareas.

Diagrama Gantt del proyecto:



Nombre	Fecha de...	Fecha de...
Estudio TensorFlow	18/09/17	8/12/17
Estudio Python	2/10/17	8/12/17
Instalación de Tensorflow y Pyth...	25/09/17	6/10/17
Estudio de material de aprendiz...	22/09/17	19/10/17
Busqueda de un tema	9/10/17	27/10/17
Estudio de la API de Twiter	30/10/17	3/11/17
Desrrollo de codigo	6/11/17	10/11/17
Busqueda de una alternativa	13/11/17	17/11/17
Estudio de la nueva aplicación	20/11/17	24/11/17
Estudio de la API de Spotify	20/11/17	1/12/17
Desarrollo del colector de datos	27/11/17	15/12/17
Desarrollo del modelo neuronal ...	4/12/17	22/12/17
Prueba y corrección del colector	18/12/17	22/12/17
Ajuste y corrección del modelo n...	25/12/17	12/01/18
Pruebas finales y presentación	15/01/18	19/01/18

Proceso de aprendizaje

Para la inmersión a esta nueva tecnología se ha utilizado el material proporcionado por el profesor complementado con material encontrado en internet.

La fase inicial del proyecto y el proceso de instalación fue guiado por la pagina oficial de TensorFlow: https://www.tensorflow.org/ y el libro: *Getting Started with TensorFlow* de Gianccarlo Zaccone. En este libro se encuentra información bien estructurada y explicada para personas que no han visto nunca esta tecnología de Google.

La siguiente parte del aprendizaje fue llevada a cabo con dos video tutoriales:

Understanding the Foundations of TensorFlow: curso dado por Janani Ravi, coofundadora de Loonycorp. Este curso está enfocado en aprender a usar esta tecnología, pero más enfocado en el tratamiento de imágenes, algo que no tiene nada que ver con la aplicación desarrollada.

Building and Deploying Applications with TensorFlow: impartido por Adam Geitegy. En este curso se puede seguir el diseño de un modelo neuronal multicapa de predicción de precios, modelo que se ajusta más a la aplicación realizada. Este curso ha servido para entender las redes neuronales y como trabaja TensorFlow con ellas.

Además de lo citado anteriormente, también ha habido un proceso de aprendizaje de la API de Spotify utilizando su web, https://developer.spotify.com/web-api/, y de las librerías utilizadas:

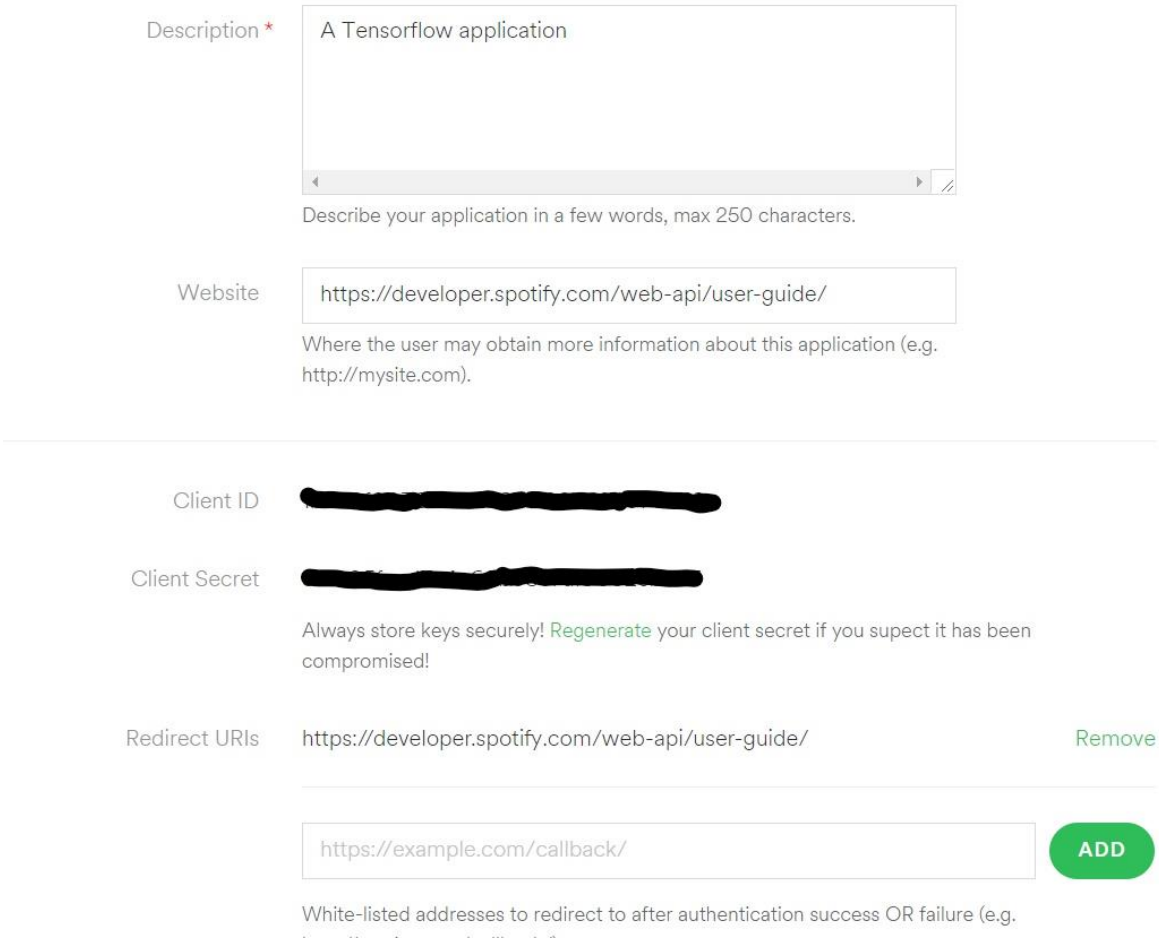
Spotipy: http://spotipy.readthedocs.io/en/latest/

Pandas: https://pandas.pydata.org/pandas-docs/stable/

La API de Spotify

Para acceder al api es necesario tener una cuenta de Spotify y crearse una aplicación. El código de Python necesita un id (Client ID) y una clave (Client Secret) para poder acceder a los datos.

Este id y esta clave tienen que ser introducidos en el código para que se establezca un token de autenticación y poder así recibir los permisos para navegar por la api.



Spotify web API: https://developer.spotify.com/web-api/

Los datos

La API de Spotify permite la obtención de una gran cantidad de datos relacionados con canciones, artistas y álbumes. Además, permite listar las canciones de una determinada playlist, que es lo que se va a usar para esta aplicación.

	A	B	C	D	E	F	G	H	I	J	
1	Column1	Column2	Column3	Column4	Column5	Column6	Column7	Column8	Column9	Column10	Colu
2		Title	Artist	Album	Popularity	acousticness	danceability	duration_ms	energy	instrumentalness	key
3	0	Amerika	Young the Giant	Home of the Strange	63	0.0141	0.625	240240	0.781	1.17e-06	0
4	1	Something To Believe In	Young the Giant	Home of the Strange	64	0.0245	0.623	228467	0.886	0.0	1
5	2	Elsewhere	Young the Giant	Home of the Strange	50	0.126	0.506	224413	0.674	0.0	5
6	3	Mr. Know-It-All	Young the Giant	Home of the Strange	57	0.0185	0.627	191747	0.871	0.0	2
7	4	Jungle Youth	Young the Giant	Home of the Strange	50	0.00538	0.565	220813	0.915	4.6e-06	5
8	5	Titus Was Born	Young the Giant	Home of the Strange	54	0.362	0.586	242627	0.452	0.000912	0
9	6	Repeat	Young the Giant	Home of the Strange	55	0.0454	0.656	185200	0.806	2.33e-05	1
10	7	Silvertongue	Young the Giant	Home of the Strange	67	0.00975	0.673	197427	0.789	0.0	0
11	8	Art Exhibit	Young the Giant	Home of the Strange	54	0.463	0.49	243640	0.48	2.27e-06	8
12	9	Nothing's Over	Young the Giant	Home of the Strange	48	0.00899	0.681	264147	0.738	0.00161	7
13	10	Home of the Strange	Young the Giant	Home of the Strange	46	0.00642	0.558	156507	0.856	0.00033	0
14	11	Heavydirtysoul	Twenty One Pilots	Blurryface	74	0.00397	0.613	234813	0.873	0.00111	7
15	12	Stressed Out	Twenty One Pilots	Blurryface	82	0.0462	0.734	202333	0.637	2.29e-05	4
16	13	Ride	Twenty One Pilots	Blurryface	82	0.00835	0.645	214507	0.713	0.0	6
17	14	Fairly Local	Twenty One Pilots	Blurryface	70	0.0396	0.715	207160	0.723	8.57e-05	6
18	15	Tear In My Heart	Twenty One Pilots	Blurryface	74	0.0189	0.655	188493	0.632	0.0	2
19	16	Lane Boy	Twenty One Pilots	Blurryface	72	0.0573	0.616	253093	0.811	1.57e-05	7
20	17	The Judge	Twenty One Pilots	Blurryface	72	0.177	0.423	297760	0.797	0.0	0
21	18	Doubt	Twenty One Pilots	Blurryface	69	0.121	0.719	191493	0.645	0.00316	2
22	19	Polarize	Twenty One Pilots	Blurryface	69	0.122	0.597	226787	0.666	0.00302	6
23	20	We Don't Believe What's On TV	Twenty One Pilots	Blurryface	68	0.23	0.547	177200	0.936	3.04e-06	9
24	21	Message Man	Twenty One Pilots	Blurryface	68	0.0897	0.715	240080	0.838	0.0	2
25	22	Hometown	Twenty One Pilots	Blurryface	67	0.0577	0.683	234947	0.753	0.00208	2
26	23	Not Today	Twenty One Pilots	Blurryface	66	0.0605	0.77	238040	0.706	0.00182	2

El objetivo es clasificar una lista de canciones en canciones que le puedan gustar al usuario o canciones que no. Para entrenar la aplicación y sepa reconocer esta clasificación se van a crear dos playlist. Una tendrá canciones positivas (le gustan) y la otra, negativas. Para ello, a estas canciones se les va a dar una variable 1 o 0, dependiendo de la clasificación. TensorFlow se va a basar en unas variables proporcionadas por la API de Spotify para realizar esta clasificación. Estos datos van a presentarse en formato csv, que es el que leerá la aplicación, y son los siguientes:

- Popularity: numero entre 0 y 100 que mide la popularidad de una canción. Esta basado en el numero de reproducciones, pero en lugar de reproducciones totales, se basa en un corto período de tiempo.
- Acousticness: una medida de 0.0 a 1.0 de si la pista es acústica. 1.0 representa una alta confianza de que la pista es acústica.

- Danceability: describe la aptitud de una pista para que esta sea bailada en función de una combinación de elementos musicales que incluyen el tempo, la estabilidad del ritmo, la fuerza del ritmo y la regularidad general. Un valor de 0.0 es menos bailable y 1.0 es más bailable.
- Duration: la duración de la pista.
- Energy: la energía es una medida de 0.0 a 1.0 y representa una medida perceptual de intensidad y actividad. Normalmente, las pistas energéticas son rápidas y ruidosas. Por ejemplo, el death metal tiene mucha energía, mientras que la pista “Prelude” de Bach tiene un puntaje bajo en la escala. Las características perceptuales que contribuyen a este atributo incluyen el rango dinámico, la sonoridad percibida, el timbre, la velocidad de inicio y la entropía general.
- Instrumentalness: predice si una pista no contiene voces. Los sonidos "Ooh" y "aah" se consideran instrumentales en este contexto. Las pistas de rap o habladas son claramente "vocales". Cuanto más cerca esté el valor de instrumentación de 1.0, mayor será la probabilidad de que la pista no contenga contenido vocal. Los valores superiores a 0.5 están destinados a representar pistas instrumentales, pero la confianza es mayor a medida que el valor se acerca a 1.0.
- Key: la clave en la que se encuentra la pista. Los enteros se asignan a los tonos usando la notación de clase de tono estándar.
- Liveness: detecta la presencia de un público en la grabación. Los valores más altos representan una mayor probabilidad de que la canción se haya reproducido en vivo. Un valor por encima de 0.8 proporciona una gran probabilidad de que la pista sea en vivo.
- Loudness: el volumen general de una pista en decibelios (dB). Los valores de sonoridad se promedian en toda la pista y son útiles para comparar el volumen relativo de las pistas. La sonoridad es la calidad de un sonido que es el principal correlato psicológico de la fuerza física (amplitud). Los valores suelen estar en un rango entre -60 y 0 db.

- Mode: el modo indica la modalidad (mayor o menor) de una pista, el tipo de escala a partir de la cual se deriva su contenido melódico. Mayor está representado por 1 y menor es 0.
- Speechiness: detecta la presencia de palabras habladas en una pista. Cuanto más se hable exclusivamente, como en una grabación, más cercano a 1.0 el valor del atributo. Los valores superiores a 0.66 describen pistas que probablemente están hechas completamente de palabras habladas. Los valores entre 0,33 y 0,66 describen pistas que pueden contener tanto música como discurso, ya sea en secciones o en capas, incluidos casos como el rap. Los valores por debajo de 0.33 probablemente representen música y otras pistas que no sean del habla.
- Tempo: el tempo general estimado de una pista en tiempos por minuto (BPM). En la terminología musical, el tempo es la velocidad o el ritmo de una pieza determinada y deriva directamente de la duración media del tiempo.
- Time_signature: la firma de tiempo global estimada de una pista. El tiempo (metro) es una convención de notación para especificar cuántos tiempos hay en cada barra (o medida).
- Valence: una medida de 0.0 a 1.0 que describe la positividad musical transmitida por una pista. Las pistas con alta valencia suenan más positivas (por ejemplo, alegre, alegre, eufórica), mientras que las pistas con baja valencia suenan más negativas (por ejemplo, triste, deprimido, enojado).
- Appreciation: valor 0 o 1 dependiendo si la pista le gusta al usuario o no.

La aplicación se va a nutrir de estas variables a la hora de ser entrenada y clasificar las canciones.

El código

El código de la aplicación esta dividido en dos partes, unidas por el mismo lenguaje de programación, Python. Esta división es meramente simbolica, pues el código inicialmente ce compone de 4 archivos .py.

La primera parte es la que trabaja con la API de Spotify y es la encargada de la recolección y depuración de datos. Es necesaria una conexión a internet y una cuenta en Spotify Developer, cuenta que es gratuita.

La otra parte trabaja con la librería TensorFlow y es la encargada de crear, entrenar y ejecutar la red neuronal utilizando los datos proporcionados por la parte anterior.

El funcionamiento de ambas partes se explica a continuación.

Funcionamiento de la aplicación.

El clasificador de música aquí descrito desde el punto de vista de programación se puede dividir en dos partes: el recolector de datos y la red neuronal.

Recolector de datos

Escrito en Python, utiliza la librería de la API de Spotify, “spotipy”, para realizar las consultas de datos contra esta API; y la librería “pandas” de Python para gestionar estos datos y transformarlos a un archivo csv.

Las consultas se realizan mediante un código autentificación que proporciona la propia API para lo cual es necesario registrarse. Todo ello es gratuito.

El recolector esta dividido en dos ficheros .py, uno se encarga de obtener los datos de entrenamiento y test, y el otro es el encargado de obtener los datos de la playlist a analizar.

Además de estos dos ficheros, hay uno adicional Playlists.py que se encarga de mostrar los ids de las playlist de un usuario, necesarios para descargar los datos utilizando los otros dos ficheros.

Los dos ficheros de recolección funcionan con las claves de usuario para la autenticación, el id de la playlist a descargar y el id del usuario que tiene esa playlist. Para modificar estos datos es necesario hacerlo a mano, en el código, ya que no hay una interfaz gráfica implementada (posibilidad de futuro).

Para estos cambios solo hay que modificar las variables que contienen esta información:

```
27  data = pd.DataFrame()
28  id_usuario = '118531625'
29  id_playlist = '21xZfbKhrSLKbJO4Aa4jG6'
```

En el caso de el fichero encargado de recolectar la plylist a analizar, SpotifyCollector.py estas variables se encuentran en las líneas 28 y 29 del código. En el código del recolector de los datos de entrenamiento se encuentran en las líneas: 30, 31, 88 y 132. En este caso, como los datos de entrenamiento se nutren de 3 playlist diferentes, son necesarios 3 ids diferentes.

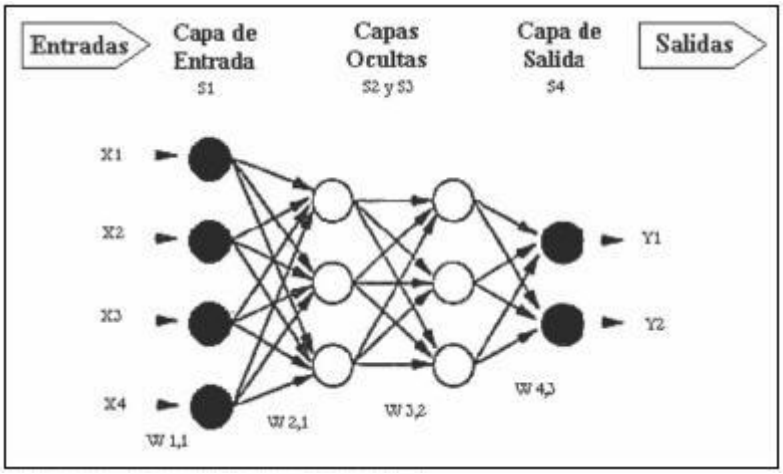
Estos ficheros son independientes de la red neuronal ya que solo es necesario ejecutarlos una vez porque se almacenan los datos en la carpeta que los contenga, datos que luego usara la red.

Red neuronal

La otra parte de la aplicación es la que utiliza TensorFlow. Para esta clasificación se ha utilizado una red neuronal multicapa que permite clasificar objetos que no son linealmente separables. En este caso se utiliza un aprendizaje supervisado en el que se le da a la red las etiquetas en las que tiene que clasificar.

Las capas pueden clasificarse en tres tipos:

- Capa de entrada: Constituida por aquellas neuronas que introducen los patrones de entrada en la red. En estas neuronas no se produce procesamiento.
- Capas ocultas: Formada por aquellas neuronas cuyas entradas provienen de capas anteriores y cuyas salidas pasan a neuronas de capas posteriores.
- Capa de salida: Neuronas cuyos valores de salida se corresponden con las salidas de toda la red.



Proceso de entrenamiento

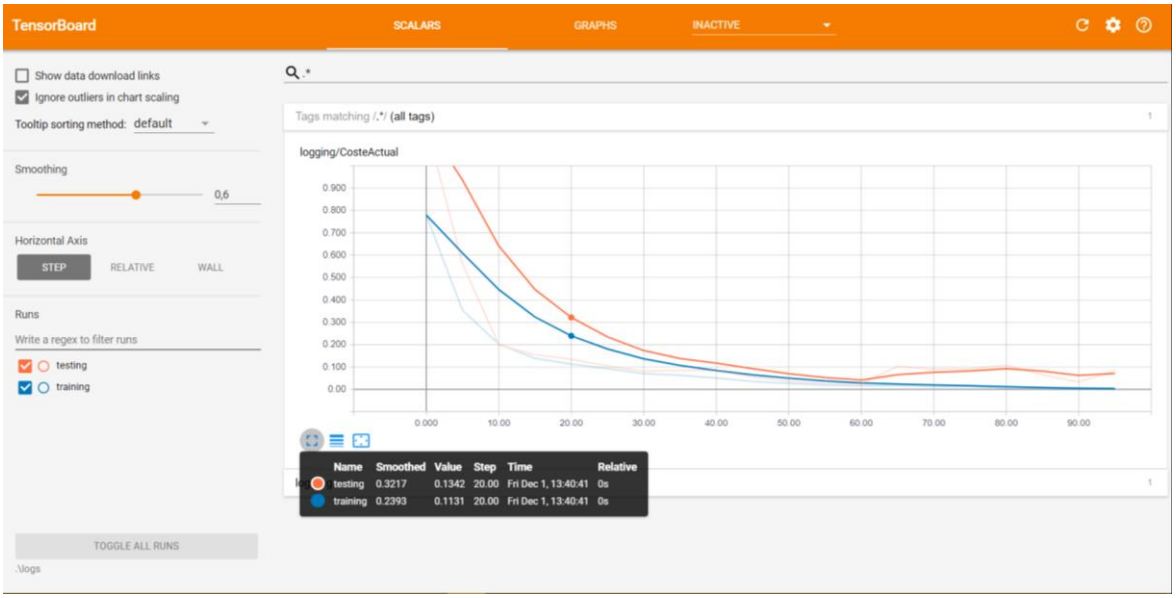
Entrenamiento inicial

El modelo inicial ha sido creado con 6 capas con distinto número de neuronas en cada una de ellas, y ha sido entrenado con aproximadamente 130 canciones.

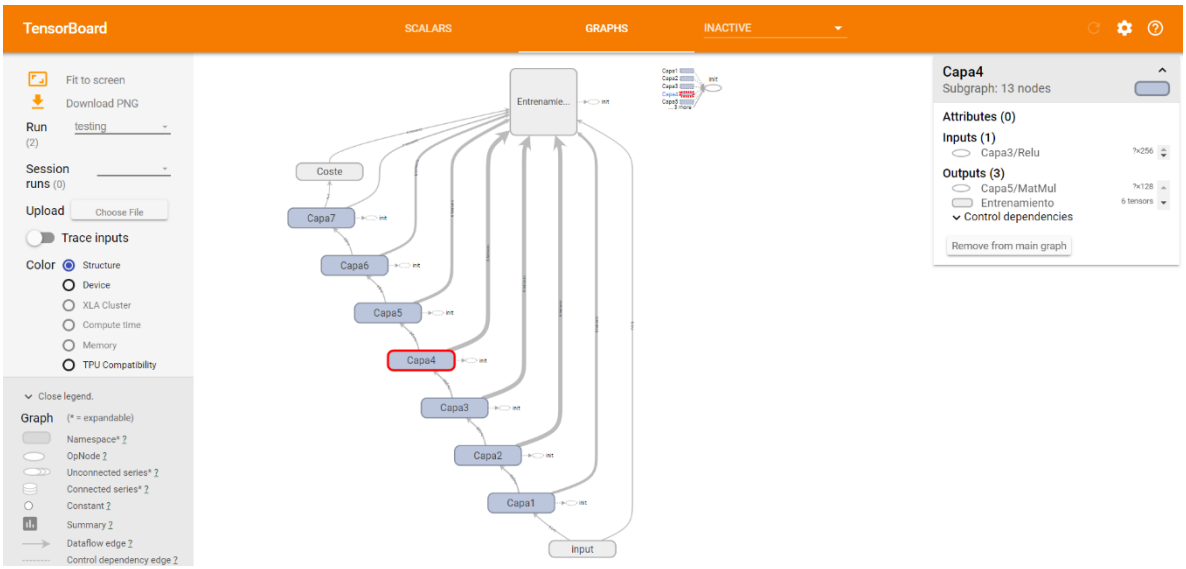
El primer resultado ha sido una predicción bastante ajustada, aunque con algún fallo que se ha de corregir posteriormente.

```
Coste de entrenamiento final: 0.03396455943584442
Coste de test finalt: 0.09072225540876389
Valoracion real: 1 --- Valoracion predecida: 0.7801579833030701
Valoracion real: 1 --- Valoracion predecida: 0.9219865202903748
Valoracion real: 1 --- Valoracion predecida: 0.9299079179763794
Valoracion real: 1 --- Valoracion predecida: 0.8123866319656372
Valoracion real: 0 --- Valoracion predecida: 0.0
Valoracion real: 0 --- Valoracion predecida: 0.0
Valoracion real: 1 --- Valoracion predecida: 0.007275670766830444
Valoracion real: 0 --- Valoracion predecida: 0.0
Valoracion real: 0 --- Valoracion predecida: 0.0
Valoracion real: 1 --- Valoracion predecida: 0.9710023999214172
Valoracion real: 0 --- Valoracion predecida: 0.0
Valoracion real: 1 --- Valoracion predecida: 1
```

Se ha reducido el coste de entrenamiento hasta valores bastante bajos, pero aún se puede ajustar más. De hecho, se puede ver que se produce algo de sobre entrenamiento que hay que corregir.



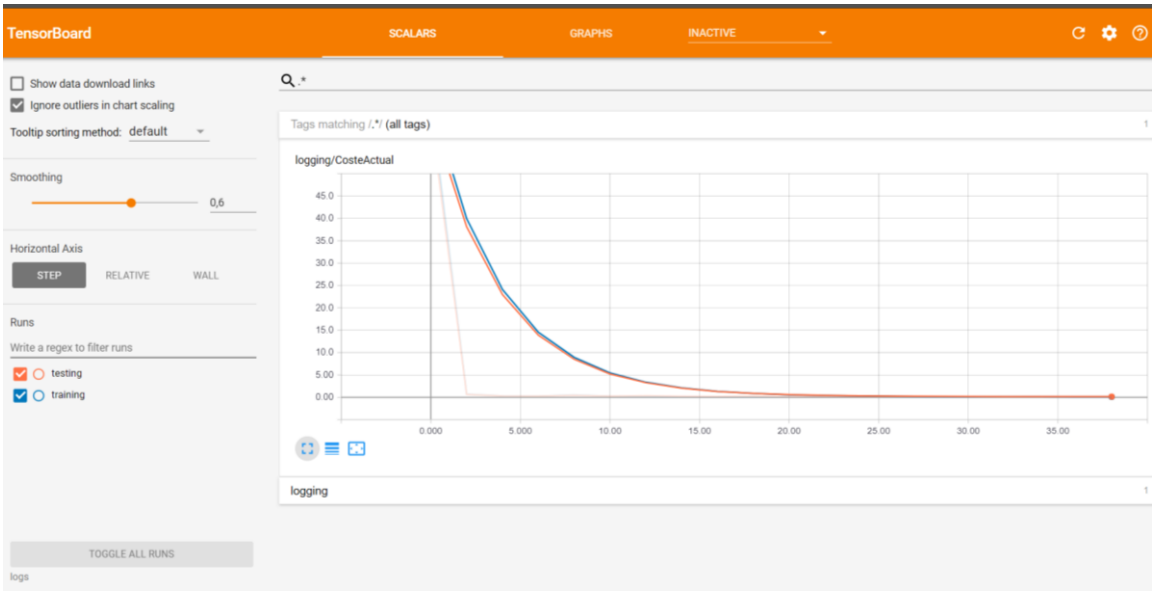
El grafo de la red neuronal sería el siguiente.



Entrenamiento después de ser ajustado

Se ha ajustado el modelo a 8 capas con un numero de neuronas decreciente, desde las 1500 de la primera, a las 8 neuronas de la última para dar un output entre 0 y 1. En este caso la predicción sigue sin ser exacta, pero se ajusta algo más a los gustos del usuario fallando 2 de 27 canciones.

La curva de entrenamiento también se ha ajustado, y esto se puede ver en la evolución del coste de entrenamiento gracias a tensorboard. Se ha corregido el sobre coste anterior, ahora la función de estrictamente decreciente.



Resultados

Tras realizar el proceso de entrenamiento es hora de aplicar la red neuronal a una playlist cualquiera para saber que canciones se ajustan a los gustos del usuario. Se ha integrado la generación de resultados en el código de la red, así que una vez entrenada comprueba la playlist que tiene asignada, y genera un archivo csv llamado ValoresPredecidos.csv.

Este archivo contiene los títulos de las canciones junto con el artista y una predicción entre 0 y 1 en función del gusto del usuario que ha entrenado la red. Cabe destacar que no todos los 0 corresponden con canciones que no le van a gustar al usuario, también un 0 se ha de asemejar con aquellas canciones que la red no tiene datos con que compararlas.

Esto se debe a que, si se entrena con unos valores determinados, un estilo de música muy concreto, a la hora de analizar una canción radicalmente distinta, la red no encuentra ninguna semejanza ni ningún patrón con que compararla. En este caso se mostrará con valores bajos.

Column1	Column2	Column3	Column4
	Title	Artist	Appreciation
0	Amerika	Young the Giant	0.71
1	Something To Believe In	Young the Giant	0.76
2	Elsewhere	Young the Giant	0.65
3	Mr. Know-It-All	Young the Giant	0.74
4	Jungle Youth	Young the Giant	0.73
5	Titus Was Born	Young the Giant	0.69
6	Repeat	Young the Giant	0.74
7	Silvertongue	Young the Giant	0.81
8	Art Exhibit	Young the Giant	0.63
9	Nothing's Over	Young the Giant	0.71
10	Home of the Strange	Young the Giant	0.72
11	Heavydirtysoul	Twenty One Pilots	0.75
12	Stressed Out	Twenty One Pilots	0.78
13	Ride	Twenty One Pilots	0.78
14	Fairly Local	Twenty One Pilots	0.74
15	Tear In My Heart	Twenty One Pilots	0.76
16	Lane Boy	Twenty One Pilots	0.79
17	The Judge	Twenty One Pilots	0.74
18	Doubt	Twenty One Pilots	0.78
19	Polarize	Twenty One Pilots	0.75
20	We Don't Believe What's On TV	Twenty One Pilots	0.77
21	Message Man	Twenty One Pilots	0.74
22	Hometown	Twenty One Pilots	0.75
23	Not Today	Twenty One Pilots	0.75
24	Goner	Twenty One Pilots	0.61
25	Ode To Sleep	Twenty One Pilots	0.76
26	Holding On To You	Twenty One Pilots	0.77
27	Migraine	Twenty One Pilots	0.79
28	House Of Gold	Twenty One Pilots	0.78

En este caso se puede ver que las canciones anteriores pertenecen a un mismo estilo de hecho, en la imagen se distinguen solo dos artistas. Con este tipo de canciones ha sido entrenada la red y por tanto la nota general supera el 0.5.

En el caso de la imagen de abajo, se han cogido canciones de todos los tipos y se puede ver como la mayoría tienen baja nota. Esta nota tan cerca del 0, como ya se ha explicado anteriormente, no quiere decir que la canción no sea del agrado del usuario, sino que pude ser que la red no tenga datos para valorar algunas canciones en concreto. En este caso les da una nota baja porque no puede asegurar que al usuario le vaya a gustar.

Column1	Column2	Column3	Column4
322	Don't You Want Me - 2002 - Remaster	The Human League	0.62
323	Big In Japan - Original	Alphaville	0.74
324	I'm So Excited	The Pointer Sisters	0.1
325	Uptown Girl - Live In Russia, 1987	Billy Joel	0.02
326	D'yer Maker	Studio 99	0.89
327	Words	Soraya	0.01
328	The Love Boat Theme	Silver Screen Symphony	0.02
329	The Sound of Silence	Simon & Garfunkel	0.01
330	I Lost You	Elvis Costello	0.02
331	ÃœBerlin	R.E.M.	0.72
332	Songbird	James Walbourne	0.03
333	Cheek To Cheek	Ella Fitzgerald	0.03
334	Sweet Caroline - Single Version	Neil Diamond	0.02
335	Don't You Just Know It	Huey "Piano" Smith	0.24
336	Hava Nagila	The Klezmer Lounge Band	0.02
337	Destiny	Zero 7	0.34
338	Summer Of '69	Bryan Adams	0.03
339	With Or Without You	U2	0.02
340	Take On Me	a-ha	0.02
341	Careless Whisper	George Michael	0.87
342	If Today Was Your Last Day	Nickelback	0.01
343	Fields Of Gold	Sting	0.02
344	Black Or White	Michael Jackson	0.02
345	Dancing in the Dark	Bruce Springsteen	0.01
346	Fireflies	Owl City	0.01
347	Luka	Suzanne Vega	0.01
348	Wind Of Change	Scorpions	0.01
349	Bitter Sweet Symphony - 2004 Digital Remaster	The Verve	0.01
350	Uptown Girl - Radio Edit	Westlife	0.83
351	Aicha	Outlandish	0.79
352	No Scrubs	TLC	0.88

Análisis DAFO

Un análisis DAFO es una herramienta sencilla de realizar y muy práctica para conocer el verdadero estado del proyecto. Su nombre proviene de las cuatro partes del mismo: Debilidades, Amenazas, Fortalezas y Oportunidades. En la gestión de proyectos, el análisis DAFO se convierte en un instrumento muy valioso para analizar el estado real del proyecto.

Fortalezas	Debilidades
• Proyecto asequible. • Software de acceso libre y gratuito. • No es necesaria mucha capacidad de computo.	• Predicción no exacta. • Necesidad de un mayor refinado. • Tecnología desconocida.
Amenazas	**Oportunidades**
• Dependencia de software de terceros. • Spotify ya tiene un sistema de recomendación. • Muy dependiente de los usuarios.	• Posibilidad de corregir las debilidades. • Introducción de nuevas variables. • Creación de una interfaz para facilitar el uso.

Lecciones aprendidas

Independientemente del resultado final del proyecto, esta claro que durante todo el proceso de realización se han tenido que adquirir ciertos conocimientos que en la mayoría de los casos no se poseían previamente.

Después de todo el trayecto y como uno de los puntos de cierre de este informe, se van a mencionar las principales lecciones que se han aprendido durante la realización del trabajo.

1. Python: uno de los leguajes de programación más versátiles y el principal pitar de este proyecto. Python ha sido necesario para todas y cada una de las partes del trabajo.

   Aunque después de haber finalizado uno no se puede considerar un experto en este lenguaje, si que ha servido para profundizar en el y, de cara al futuro, ser capaz de desenvolverse y resolver problemas que se planteen en dicho lenguaje.
2. Tensorflow: el otro pitar del proyecto y el software sobre el que se ha basado todo el trabajo. Gracias a esta experiencia se ha conocido y explorado esta librería de Google que tanto facilita el trabajo de aprendizaje de máquina, concretamente el área de Deep Learning.

   Al igual que con Python, después de haber realizado este trabajo no podemos considerarnos expertos puesto que esta librería ofrece infinidades de posibilidades, pero si ha servido para conocer la tecnología y ser capaces de crear una aplicación con ella.
3. Api de Spotify y Spotipy: Spotify ofrece gran cantidad de información relacionada con su servicio de música en streaming, información que puede ser obtenida de diversas formas. Una de ellas es a través de la librería Spotipy de Python, la cual se ha tenido que conocer cómo funciona, cuáles son sus métodos y sus clases, etc. Como ya se ha mencionado anteriormente, es necesario tener una cuenta de Spotify para interaccionar

con la API, y esta interacción es llevada a cabo por la librería, por tanto, es necesario conocer ambas partes.

4. Pandas: es un paquete de Python que proporciona estructuras de datos similares a los dataframes de R. Es útil para todo el tema de tratamiento y manejo de datos, además facilita su conversión a diferentes tipos. Al tener tantas ventajas, el uso de pandas simplificaría mucho los datos, a pesar de que ello implicarse aprender cómo funciona desde cero.
5. Entornos virtuales (Anaconda): también se ha aprendido a manejar el entorno de Anaconda, un software que incorpora Python y que facilita la instalación de paquetes. Este software se usó como primera opción, pero después de un tiempo se descartó, y se instaló Python en raíz.
6. Api de Twitter y Tweepy: como añadido otra de las cosas aprendidas durante el proyecto es el manejo de la API de Twitter. Esto de debe a que el proyecto inicial comprendía un análisis de sentimiento usando Twitter, pero una vez encontrados varios problemas se cambió por el trabajo de predicción actual. Al igual que la API de Spotify, esta también dispone de una librería de Python para su manejo, Tweepy.

Posibilidades de futuro

A nivel de aplicación

Una de las posibilidades a este nivel seria añadir una variable que corresponda con el género de las canciones para que la red neuronal la tuviese en cuenta. Actualmente Spotify no clasifica sus canciones en géneros, sino que aplica esta clasificación a los álbumes o a los artistas, pero nunca asociado a un título de una canción.

También en un futuro se pretende crear una aplicación más grafica visualmente y/o la posibilidad de crear una aplicación web.

A nivel empresarial

Posibilidad de trabajar con Spotify para mejorar su sistema de recomendación, lo cual es difícil porque el sistema actual es bastante completo, pero no imposible.

https://www.xataka.com/aplicaciones/como-acierta-tanto-spotify-en-su-discovery-semanal

Enlaces externos

GitHub: https://github.com/Ramonch/SpotifyAnalisis

Página oficial de TensorFlow: https://www.tensorflow.org/

API de Spotify: https://developer.spotify.com/web-api/

Documentación pandas: https://pandas.pydata.org/pandas-docs/stable/

Documentación spotipy: http://spotipy.readthedocs.io/en/latest/