## NLP para la clasificación de canciones

En este artículo vamos a contar cómo podemos utilizar el Machine Learning para desarrollar un modelo que sea capaz de entender la letra de distintas canciones, así como su posterior clasificación en diversas categorías. En concreto, utilizaremos diversas tecnologías, como la frecuencia de término y la frecuencia inversa de documento, situadas dentro del Procesamiento del Lenguaje Natural.

## Construcción del modelo

En esta parte, vamos a describir el proceso de creación de nuestro modelo.

En primer lugar, partimos de un dataset formado por las urls que nos direccionaban a las letras de las canciones y por el target multietiqueta que nos habla de las distintas categorías a las que las canciones pueden pertenecer.

Como bien es sabido, lo primero que debemos hacer es limpiar y preparar nuestro datos. Para ello, en este primer paso trabajamos con las librerías preprocess_kgptalkie(kgp) y Spacy.

Kgp se trata de una librería especializada en el preprocesamiento de textos. Con ella, nos centramos en la limpieza de nuestros textos, pues como hemos obtenido las letras de las canciones de urls lo más común es que vengan con información que no nos es relevante. Ayudandonos de esta librería realizamos diversos filtrados como la eliminación de caracteres especiales, de emails, de urls, de html tags, etc.

Por otro lado, contamos con la ayuda de Spacy, que es una librería muy amplia centrada en todos los aspectos del NLP. En nuestro caso, la utilizamos para eliminar los stopwords de nuestros datos. Las stopwords son las palabras que aparecen en prácticamente todos los textos en repetidas ocasiones y no aportan un contexto ni relevancia importante en los textos. Un ejemplo son las preposiciones, conectores…

Una vez hemos tenido los datos ya preprocesados hemos pasado a la preparación de estos para la construcción de nuestro modelo. En este caso hemos utilizado la estrategia TFIDF.

La estrategia frecuencia de término-frecuencia inversa o TFIDF, es una medida numérica que expresa cuán relevante es una palabra para un documento en una colección.

La frecuencia de término(Term Frequency o TF) es simplemente la relación entre el número de palabras presentes en un documento y la longitud del mismo.

> TF(t) = (Número de veces que aparece el término t en el documento)/(Número total de términos en el documento)

La frecuencia inversa reduce las palabras que aparecen mucho en el documento

> IDF=log(N/ TF(t)) , donde N es el numero de documentos que contienen el término t

Por último, para calcular TFIDF, debemos hacer:

TFIDF= TF * IDF

Ayudándonos ahora de la librería sklearn, en concreto de TFidfVectorizer. Hemos pasado de tener un dataset con palabras, con lo que no puede trabajar ningún modelo, a tener un dataset formado por un target y una matriz de valores numéricos. El problema es que si trabajamos con textos grandes como es nuestro caso, nos crea unos datos con dimensiones muy grandes. En particular pasamos a tener un dataset formado por 370 filas y 428732 atributos. Para solucionar este problema, tenemos que optar por la reducción del dataset mediante análisis de componentes principales.

Debemos pasar de describir nuestro conjunto de datos en términos de nuevas variables no correlacionadas. Para ello utilizamos una serie de pruebas, utilizando de nuevo sklearn, hasta que vimos que podíamos realizar una reducción de dimensiones de 428732 atributos a un rango de 220–270 atributos (dependiendo del caso) explicando el 90% de la varianza del documento.

Una vez tenemos los datos preparados pasamos a la selección del modelo, para ello utilizamos spacy, sklearn y tensorflow. En concreto, utlizamos Support Vector Machine, Random Forest, K-Nearest Neighbours, construimos una red neuronal clasificadora y word2vectors de Spacy. Una vez hechas las pruebas optamos por la optimización del modelo que nos aportaba mejores resultados, Support Vector Machine.

El primer paso era la optimización de TFIDF, para ellos utilizamos Cross Validation. En concreto utlizamos GridSearchCV de sklearn, modificando los parámetros más importantes y realizando una clasificación del modelo inicial de SVM.

Una vez obtenido el mejor TFIDF, aplicamos a los datos un PCA que nos el dataset a 220 atributos con una varianza explicativa del 90%.

El siguiente paso fue la optimización del SVM, en concreto utilizamos un OneVsRestClassifier con el estimador LinearSVC. En este caso, también utilizamos Cross Validation.

## Resultados

Una vez hemos tenido entrenado y optimizado el modelo. Hemos podido realizar la clasificación multietiqueta de nuestras canciones. En concreto el modelo es capaz de descargar los datos de una url externa, limpiarlos y transformarlos a su gusto para su posterior clasificación en una de las posibles 8 categorías.

Contábamos con un dataset de 370 canciones y hemos realizado la clasificación multietiqueta con un score del 53.38%, lo cual teniendo en cuenta la dificultad del proceso y los pocos datos que teníamos nos damos cuenta del buen tratado de los datos y la buena optimización de los procesos.

## Otras aplicaciones

Viendo este pequeño proyecto y los resultados que hemos llegado a obtener nos planteamos qué otros campos podríamos utilizar dichos conocimientos y las posiblidades son inmensas.

Podríamos ser capaces de detectar sentimientos y emociones, como detectar discursos de odio en tweets. Se pueden generar chats de voz como cortana o siri. También potentes traductores como es el caso de DeepL. Se puede optimizar la búsqueda de empleados mediante la extracción de las partes importantes de un currículum.

Y esto son solo unos ejemplos, realmente es un campo que se puede explotar mucho, pues los límites aún no han sido encontrados.