Transgender Vocoder



Memoria Trabajo Fin de Máster: “Transvocoder”

Autor: Francisco Romero García

https://github.com/powermates/Transvocoder

31-08-2020

# Introducción:

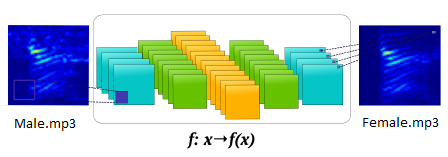
El presente proyecto trata de demostrar la capacidad de las redes convolucionales en la transformación de voces.

En un primer planteamiento del proyecto se había pensado como uso para mejorar la legibilidad de voz de personas laringectomizadas y se enfocó en esa dirección, pero la falta de datos o corpus convertía esta tarea como inadecuada para este proyecto.

Posteriormente se decidió hacer un cambio en el planteamiento y hacer un conversor de voz de masculina a femenina, ya que está demostrado que las diferencias entre voces de distinto género no es sólo de pitch o entonación y que disponen de muchas otras características como reverberación, armónicos, etc.

Se intenta reproducir estas características utilizando Deep Learning, y aunque es un problema resuelto actualmente, las redes que hacen tareas de este tipo son de una complejidad bastante grande y requieren de una capacidad de proceso enorme y costosa.

El objetivo de este proyecto tiene un planteamiento simple y didáctico independiente del resultado, con la intención de que la creación de datos (predicción) sea instantánea. Para ello se realizará el entrenamiento de un modelo Encoder-Decoder con pares de muestras masculino vs femenino para conseguir el resultado.



# Descripción de los datos:

Se ha utilizado Mozilla Common Voice como fuente de datos de voz con el dataset de lenguaje español.

<https://commonvoice.mozilla.org/es/datasets>

Enlace directo (jun-2020):

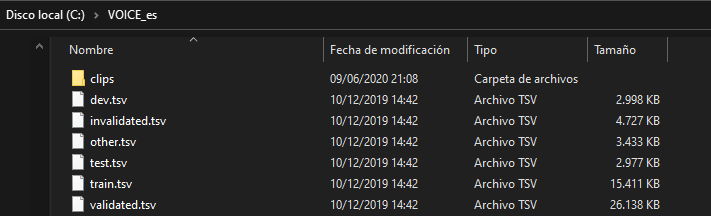
<https://voice-prod-bundler-ee1969a6ce8178826482b88e843c335139bd3fb4.s3.amazonaws.com/cv-corpus-5.1-2020-06-22/es.tar.gz>

(13GB)

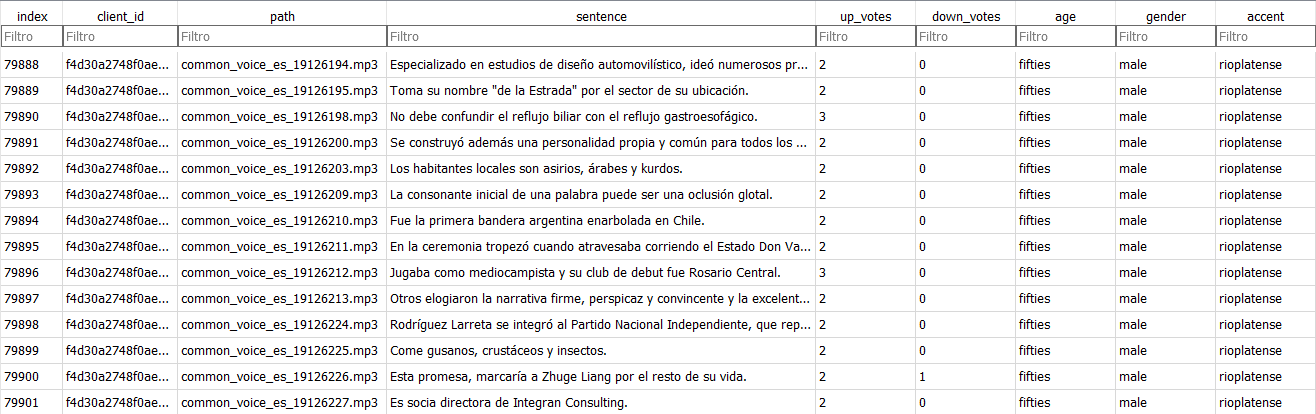
Common Voice es un proyecto de crowdsourcing iniciado por Mozilla para crear una base de datos gratuita para software de reconocimiento de voz. El proyecto cuenta con el apoyo de voluntarios que registran frases de muestra con un micrófono y revisan las grabaciones de otros usuarios.

Esto significa que gran parte del corpus no está revisado y podría contener grabaciones erróneas o incompletas. Como el problema a afrontar es bastante delicado se hace imprescindible que las muestras sean significativas y fiables.

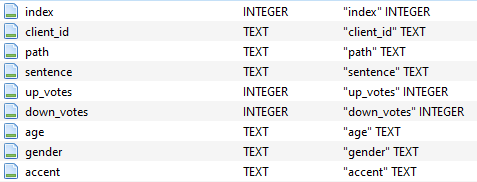
Afortunadamente, en el descargable se suministran varias hojas con metadatos:



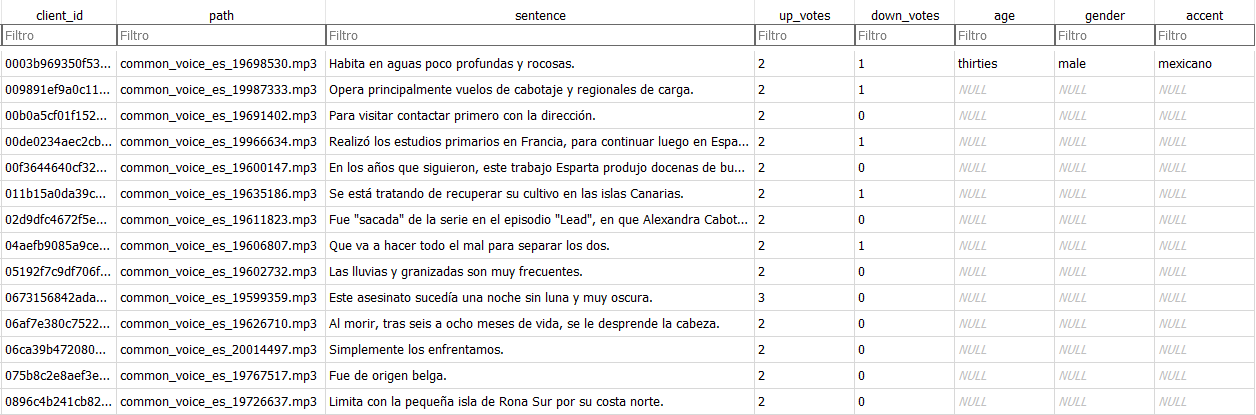
Nos centraremos en Validated.TSV que es una simple hoja CSV tabular de 112.126 filas.



De estos, describimos los importantes:



* Client\_id es un identificador del usuario autor de la voz
* Path indica el nombre del fichero.mp3 en dicha fila
* Sentence es la frase en español utilizada como referencia. Para el presente proyecto no se utilizan Word embeddings con lo que el contenido es irrelevante, sólo el hecho de identificar la frase como tal.
* Up\_votes y Down\_votes indica el número de votos positivos y negativos en la validación. En Validated.TSV sólo aparecen registros con votos positivos>1 y votos negativos <2, que es el objetivo que buscamos.
* Age, gender, accent: Son autoexplicativos, los usaremos para hacer diversas consultas que nos permitan vincular con más precisión las muestras:



Como se puede observar no siempre están rellenos estos datos, con lo que las pruebas más precisas son escasas y nos enfrentamos con la primera dificultad del proyecto.

Para ayudar a realizar consultas y experimentación con el corpus se ha añadido un proceso de conversión de la hoja a base de datos Sqlite, lo cual permite también hacer las consultas mucho más rápido ya que son bastantes líneas las que se someterán a correspondencia cruzada.

Tratamiento de las muestras:

Los datos de las muestras están en formato mp3, y se han cargado de la siguiente forma:

* Sampleado a 11khz, suficiente para voz, y que facilita el entrenamiento del modelo.
* Debido a la presencia de muestras anormalmente largas se ha puesto un tope de longitud de 100.000 samples por muestra (más de 10 segundos), con lo que descartamos las muestras de este tamaño.
* Debido a la naturaleza encoder-decoder del problema todas las muestras deben tener la misma longitud, con lo que se ha añadido un padding de ceros a la derecha de la muestra en formato bruto que es un silencio total, hasta ocupar los 100.000 samples.
* Cálculo de la Short Discrete Fourier Transform sobre ventanas a lo largo del audio y conversión a 128 MELs con la herramienta Librosa. Esto convierte los datos de 1 dimensión de 100.000x1, a 2 dimensiones más entendibles visualmente de tamaño fijo 128x196.
* Escalado de los datos MEL a la escala logarítmica dentro del rango audible por el ser humano (-80...+80 Decibelios)
* Normalización dentro de la escala -1…+1 para poder ser usado con Activación Tanh y Loss MSE.
* Debido a la lentitud de carga de los mp3 (1 seg por archivo), muy superior incluso a la del preprocesamiento en sí, se han volcado todos los datos como arrays numpy y guardados a disco. (Se incluyen dos muestras pequeñas en el repositorio (XAA.npy y YAA.npy respectivamente para X e Y) para facilitar la ejecución del proyecto)

# Metodología:

Se ha observado que en multitud de paper sobre modelos de audio se han obtenido buenos resultados en modelos de clasificación y de audio denoise (que es generativo), el uso de redes convolucionales 2d utilizando la escala MEL.

Existen un modelo (Wavenet) que predice la onda en formato bruto con la máxima calidad, pero el modelo es bastante complejo y la velocidad de predicción lo convierte en una mera prueba de concepto (horas de procesamiento para obtener uno o dos segundos de audio).

Para resolver la incógnita inherente a la reconstrucción del audio, que es la segunda dificultad del proyecto, se hicieron pruebas con modelos Autoencoder basados en redes convolucionales de 1 y 2 dimensiones, LSTM/GRU, Dense…, con distintos grados de profundidad para calibrar la escala MEL utilizada, entre 2 y 2048 MELS, para obtener un equilibrio entre complejidad del modelo y calidad del audio resultante.

De las pruebas de reconstrucción (misma voz en la entrada y reconstrucción de la misma en la salida) se consiguieron los siguientes resultados (se ha utilizado una voz femenina en inglés para observar mejor los detalles en altas frecuencias):

* A partir de 1024 la reconstrucción MEL-to-Audio es perfecta pero la librería Librosa avisa del hecho de que se encuentran muchas escalas MEL totalmente vacías, y hace falta un modelo autoencoder con cientos de filtros, demasiado lento y complejo.
* Se encontró una calidad casi perfecta al utilizar 512 mel (a pesar de que en esta prueba el audio no se normalizó y sonó distorsionado):



* Con menos de 32 mels pero el resultado suena robótico:



* Finalmente se ha utilizado 128 mels como el equilibrio perfecto entre modelo ligero, y voz no robotizada sin muchos artefactos audibles:



Llegado a este punto se toma la decisión firme de utilizar 128 mels, y una arquitectura convolucional, con lo que hay que reemplazar el Autoencoder por un modelo Encoder-Decoder adecuado. El Autoencoder solo ha servido para generar y validar una sola muestra (ya que sólo tiene que comprimir una representación de un objeto), y el objetivo es un modelo grande que aprenda y generalice con cualquier tipo de voz a partir de muchas.

Con los primeros resultados se observó la dificultad del modelo para aprender, siendo necesaria la inclusión de más filtros y más capas, pero llegada a una complejidad dada no mejoraba, encontrando la tercera dificultad del proyecto, que era encontrar el modelo adecuado.

Se optó por investigar en multitud de papers de tratamiento de imágenes y de tratamiento de sonido y se probaron modelos ya hechos adaptados a esta tarea.

Tras una decena de modelos probados, los mejores resultaros se empezaron a obtener con modelos adaptados desde implementaciones diversas de ResNet y DenseNet. Aparentemente el problema requiere de una arquitectura profunda (para almacenar la capacidad de la red), ayudado de conexiones residuales o conexiones directas.

El problema es que estos modelos tienen el objetivo final de clasificar y la utilización de capas pooling hacen perder detalles importantes finales para simplificar y acelerar el modelo, con lo que estas arquitecturas no reconstruyen de forma fiel el objetivo.

Finalmente, tras lo aprendido en todas las pruebas de modelos y en los requisitos que debía tener la red, se encontró un paper bastante interesante, “A FULLY CONVOLUTIONAL NEURAL NETWORK FOR SPEECH ENHANCEMENT” con un tipo de red denominada “Cascaded Redundant Convolutional Encoder-Decoder” o R-CED con lo que se obtuvieron varias implementaciones y se adaptaron al problema en cuestión:



Este paper de 2016 describe un modelo encoder-decoder para la reducción de sonido, pero la reconstrucción es autoregresiva en pequeños bloques.

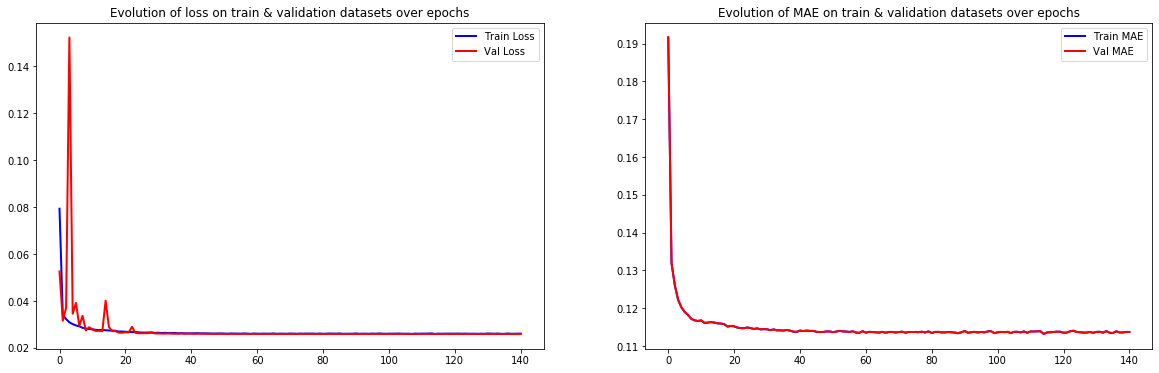
El problema de nuestro modelo es que es imprescindible cargar el audio completo de la entrada y de la salida ya que el corpus de voces no es paralelo, es decir, no está alineado, con lo que no se puede entrenar con pequeños trozos en la entrada y la salida, que es la cuarta dificultad del proyecto.

Se ha optado por respetar las especificaciones del paper en lo relativo al tamaño de los filtros, tasa de aprendizaje, regularización L2 para evitar overfitting, etc.

El entrenamiento incluso con el corpus de grande de 7000 muestras no llega a durar mucho más de una hora, ya que se ha optimizado el modelo y la entrada de datos.

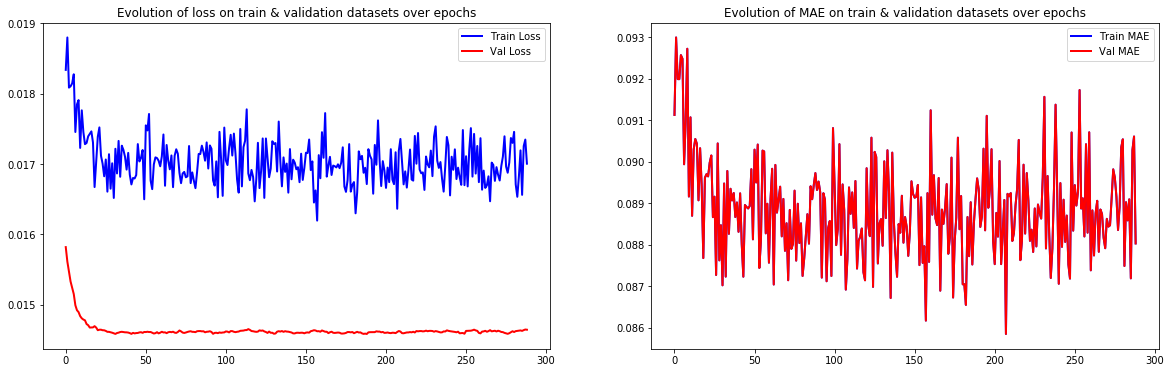
Se ha usado una GPU nVidia GTX 980TI de 6 gigas de memoria y un tamaño de batch de 5, ya que valores cercanos a 10 hacen fallar el modelo por out of memory.

Se ha entrenado la red con 7000 muestras hombre vs mujer sin filtros y 140 epochs, utilizando ReduceLROnPlateau según recomendación del paper (LR/2), y EarlyStopping :



Sin embargo, el resultado al generar el wav no es legible.

Se ha utilizado como pretraining el resultado de esta red reutilizando los pesos de la misma, pero con otro dataset más pequeño haciendo correspondencia entre genero, edad, y acento. Automáticamente la accuracy ha mejorado de golpe y se ha tuneado bajando más aún el learning rate…



Pero el resultado no es aún legible (aviso, audio estridente) :



# Resumen de resultados.

Las cuatro dificultades principales del proyecto han sido:

1. Fuente de datos: Solucionado, utilizado Mozilla Commonvoice
2. Reconstrucción de datos: Solucionado, utilizado 128-MEL y convolucionales 2D.
3. Encontrar el modelo adecuado: Parcial, modelo RCED
4. Falta de alineación de los datos: No solucionado, el problema de la fuente de datos solo se podría solucionar cambiando de fuente de datos, ya que cambiar la misma produciría efectos de distorsión en las mismas.

El resultado no es el esperado.

# Conclusiones.

Probablemente la falta de consecución del objetivo final se deba a una mala alineación de los datos, unido a una arquitectura insuficiente para la resolución de este problema.

Una buena idea sería utilizar algún generador tipo GAN capaz de reconstruir los datos de forma crítica.

Actualmente el resultado no es favorable, pero se considera de utilidad didáctica.

# Manual de usuario

Requisitos:

* Librerias: Tensorflow 2, Librosa, SQLite
* Programas: FFMPEG/SDL o compatibles instalada en el sistema operativo, para poder cargar archivos mp3.

Pasos:

* Descargar el dataset de Mozilla Commonvoice y guardarlo en una carpeta local. Es muy grande, son 13 gigabytes y 100.000 archivos con lo que se aconseja un disco duro rápido.
* Eso crea una carpeta llamada VOICE\_es con todo el contenido.
* Posteriormente hay que editar el primer fichero 0\_GenerarDatabase.py para corregir dicha ruta
* Elegir en 1\_Preprocesado.py la SQL y dataframe que queremos utilizar para generar los datos. Estas consultas están comentadas en el código.

Entonces se ejecuta:

0\_GenerarDatabase.py

1\_Preprocesado.py

2\_Entrenamiento\_Test\_RCED.py

Para la repetición solo habría que ejecutar #2 ya que los anteriores ejecutables guardan a disco los arrays numpy con las ondas preprocesadas y normalizadas listas para usar.

Los correspondientes a las 7000 muestras ocupan 600 megabytes cada array, con lo que no se han subido a Github, pero sí se han subido unos pequeños (XAA.npy YAA.npy) para poder probar sin el corpus.

En el caso de 2.py hay que editarlo y cambiar el nombre del archivo de entrada en la función predictTest() male.mp3 o usar uno propio de menos de 10 segundos de duración.

## Referencias:

* “A FULLY CONVOLUTIONAL NEURAL NETWORK FOR SPEECH ENHANCEMENT” (2016), Se Rim Park, Jin Won Lee