Generación de melodía con RNN

Lucas Steffanutto
lucasste33@gmail.com
Universidad de Buenos Aires
Redes Neuronales - Facultad de Ingeniería - 2020

Abstract

La música es un arte auditivo producido por el ser humano. Al principio se hacía con instrumentos reales mientras que hoy la mayoría de los temas son realizados por unq computadora con una Digital Audio Workstation (DAW). Sin embargo, una constante que puede producirse en el proceso de creación musical es la falta de inspiración al momento de crear una melodía, por ejemplo. En este trabajo, veremos como un músico podría adaptar un modelo de red neuronal como herramienta inteligente por sus creaciones. El objetivo es ver si podemos componer la parte melódica de un tema con Inteligencia Artificial o si, al menos, podemos servirnos de la misma como asistente durante una composición.

1 Introducción

La Inteligencia Artificial, que al principio resolvía problemas simples de álgebra, constituye hoy una ayuda en la vida real, realizando tareas cada vez más complejas, como traducciones de texto, reconocimiento de voz, visión por computadora, conducción de un vehículo, diagnóstico médico, etc. Estos avances fueron posibles con el desarrollo de las técnicas de Machine Learning (pre-training, redes convolucionales, LSTM) pero también con el crecimiento de las bases de datos y de la potencia de cálculo de las computadoras (Ley de Moore). Luego, los primeros éxitos en las empresas del sector de tecnología como Google o Facebook contribuyeron a la expansión de las investigaciones en este campo. Ahora la Inteligencia Artificial y el Machine Learning se democratizaron tanto que muchas fuentes llenan la Internet y son accesibles a todo el mundo.

Algunos sistemas inteligentes pueden sobrepasar a los humanos en varias tareas complejas, como en los juegos de ajedrez y de go, por ejemplo. Todavía no parece aplicarse a la esfera del arte en la parte de creación en sentido estricto. Sin embargo, algunos algoritmos pueden aprender un estilo de arte y después hacer un *transferencia de estilo* en un fotografía, por ejemplo. Musicalmente, algunos proyectos pueden hacerlo igual y componer, por ejemplo, una pieza imitando el estilo de Bach (D.Cope, "*Experiments in Music Intelligence*.", 1987).

Aunque las estructuras de Machine Learning sean más complejas en el arte musical, porque deben funcionar de manera dinámica, el proyecto *Magenta* de Google propuso su modelo Melody-RNN para generar una melodía. Éste consiste en un código open-source en Python, usando Tensorflow para crear música o arte con Machine Learning.

El objetivo de este trabajo es mostrar como, utilizando un modelo s*tate of the art* de inteligencia artificial, es posible generar material musical que podrá servirle a un humano para crear una pieza. Para eso, se usará un algoritmo de Deep Learning (Melody-RNN, Google Magenta, 2018 *[9]*), para generar una melodía desde cero, o desde un comienzo de melodía dado en input al modelo. A continuación se explicá la teoría de este modelo y, luego, su experimentación para generar música.

2 Teoría

2.1 Recurrent Neural Network (RNN)

Este tipo de red es similar a una red *feed-forward* pero posee neuronas con bucles de retro-alimentación. Eso permite tratar información secuencial de tamaño variable. Con estos tipos de conexiones, nos aseguramos la persistencia de los datos previos y podemos calcular y producir un output en función de lo que pasó antes a través de la red. De esta manera, resolvemos el problema de las redes neuronales tradicionales, donde la información se mueve en una sola dirección.

Este funcionamiento es similar a nuestro cerebro: cuando estamos leyendo, por ejemplo, entendemos cada palabra basándonos en las anteriores del mismo texto. Por eso, las redes RNN son muy usadas para tratar tareas donde tenemos que tomar en cuenta los datos pasados.

Entonces, si representamos una red de este tipo (***A***, *Fig1*), tenemos una salida normal h_t y dos entradas: una normal x_t y una que viene de la conexión retroactiva de las neuronas que conserva la información anterior para usarla de nuevo durante el próximo cálculo del estado de la red. Sería como si la red fuese equivalente a una cadena de diferentes copias de la misma. Si simplificamos la estructura, desarollándola como si fuera una red feed-forward, obtendremos la siguiente figura:

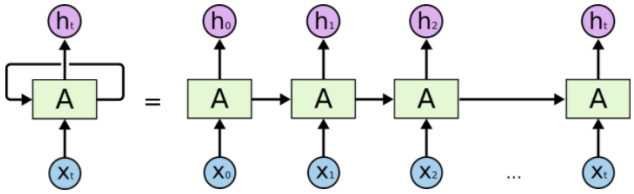

Fig.1: Esquema de una red RNN desarrollada [1]

En presente trabajo, tomaremos el caso de la generación de un contenido musical, por lo cual nuestra red RNN producirá secuencias temporales y predirá las siguientes notas en una melodía, basándose en su entrenamiento y en las notas producidas.

Por otro lado, este tipo de red se entrena con la técnica clásica de *backpropagation*, usando el *gradiente descendiente*. Pero en la práctica, sin embargo, en la fase de entrenamiento de una RNN, puede aparecer el problema de desvanecimiento de gradiente:

«El problema es que, en algunos casos, el gradiente se irá desvaneciendo a valores muy pequeños, impidiendo eficazmente el peso de cambiar su valor. En el peor caso, esto puede impedir que la red neuronal continúe su entrenamiento»[2]

Es decir, podemos encontrar dificultades cuando propagamos una error a través del tiempo, porque la recurrencia entrena multiplicaciones repetitivas y puede conducir a amplificar o minimizar los efectos (*3.1 EXPONENTIALLY DECAYING ERROR [3]*).

2.2 Long Short Term Memory (LSTM)

La solución más común para resolver este problema es usar una red Long Short Term Memory*[2].* Este método se basa también en la del *gradiente descendiente*, pero truncándolo y controlando el flujo de errores, con puertas con operaciones lógicas, que permiten acceder o no a un flujo de errores recurrentes.

«In other words, each LSTM block learns how to maintain its memory as a function of its input in order to minimize loss»[3]

Hoy, la arquitectura de Long Short Term Memory es uno de los estándares para las redes neuronales recurrentes. Hay diferentes arquitecturas de neuronas para una red LSTM pero, conceptualmente, la básica está constituida como en la (*Fig2*), con una *memory cell* que la conectamos a una neurona lineal central.

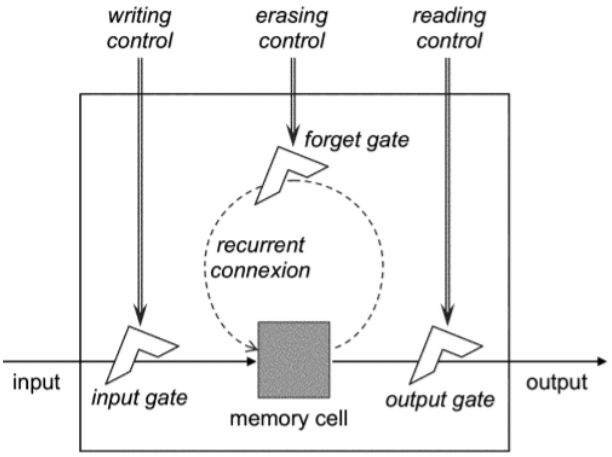

Fig.2: Arquitectura conceptual de una LSTM [4]

Para modificar sus valores durante el aprendizaje, una *memory cell* usa tres diferentes puertas, moduladas por parámetros de pesos, que pueden abrirse o cerrarse en función de las acciones: leer, escribir o borrar sus valores durante el proceso de *backpropagation*. Otros esquemas más complejos de la estructura de una LSTM se encuentran en el paper *[3].*

2.3 Attention mechanisme

Este mecanismo (*Fig4)* que vamos a usar con el modelo MelodyRNN permite enseñarle a la red estructuras más largas. Con él, la red puede acceder a una información anterior sin tener que almacenarla en una célula de estado de RNN. En este modelo se usa una cell de RNN que es una LSTM. El método de *Attention* usado en el modelo de *Magenta* proviene del paper *[6].* En el mismo, el mecanismo consiste en un *encoder-decoder* RNN, y usa *Attention* para verificar todas las salidas del encoder durante cada paso de la decodificación. En la versión *AttentionRNN* de *Magenta*, donde no hay un *encoder-decoder*, miramos sólo las salidas de

los *n* últimos pasos cuando generamos una salida para el paso actual. La innovación del mecanismo es que la red va a focalizarse sobre los eventos pasados más importantes, de manera similar a la que funciona nuestro cerebro, es decir, ponderando los miles de estímulos que recibe. Entonces, la red realiza los siguientes pasos:

i. $u_i^t = v^T \tanh(W_1' h_i + W_2' c_t)$

ii. $a_i^t = \mathrm{softmax}(u_i^t)$

iii. $h_t' = \sum_{i=t-n}^{t-1} a_i^t h_i$

Fig.3: Los tres pasos del mecanismo de AttentionRNN [7]

v, ***W1***, ***W2*** son los parámetros que el modelo está aprendiendo. Los h_i corresponden a las salidas del RNN de los *n* pasos previos y el c_t es el estado actual de la *memory cell* RNN del paso actual.

i. Con estos parámetros, calculamos un vector u_i^t de dimensión [1xn] que se corresponde a un valor para cada uno de los *n* pasos previos. Estos valores definen cuanta atención debe recibir cada paso anterior al momento de calcular el paso actual.

ii. Después, normalizamos estos valores con la función *softmax* con el objetivo de generar un vector de máscara a_i^t para saber si tomamos en cuenta un estado anterior o no.

iii. Luego, multiplicamos las salidas de los *n* pasos previos con el vector de máscara y sumamos las valores juntos para obtener h_t'.

Finalmente, concatenamos h_t' con la salida de la red RNN del paso actual y aplicamos un layer lineal para obtener la nueva salida del paso actual. A veces, en el modelo de *Magenta*, esta salida obtenida es también usada para alimentar la próxima entrada de la red para calcular el siguiente paso. Entonces, el mecanismo *AttentionRNN* afecta tanto la información saliente de la neurona RNN como la información que está inyectada en la misma.

El vector final es una combinación ponderada de cada uno de los últimos pasos juntos: cada uno va a contribuir con su importancia para el paso actual y a ayudar al modelo a aprender *longer-term dependencies* sin almacenar información.

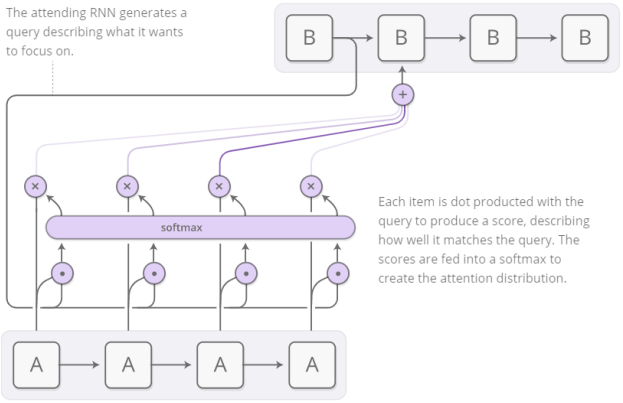

Fig.4: Esquema del funcionamiento del mecanismo Attention [5]

3 Experimento

3.1 Objetivo

El objetivo principal de este trabajo es generar una ayuda a la persona que quiera producir un tema y que no tenga muchas nociones de teoría musical, o que no tenga inspiración en el momento. Entonces vamos a generar una melodía, adaptando el modelo, para que el usuario pueda influenciar el resultado gracias a tres variables: la aleatoriedad o *temperatura* (*T*), el *quarter per minutes* (*qpm*) y el *input* que damos al modelo para la generación.

- La *temperatura (T)* va a definir el nivel de aleatoriedad y de originalidad de la melodía generada, actuando sobre la redundancia de las notas generadas.
- El *qpm* es el número promedio de notas generadas por cada minuto.
- El *input* es lo que designa lo que pasamos al modelo. Tenemos dos opciones: o usamos un archivo MIDI de un mínimo de una nota, para que el modelo genere las notas que siguen, o no ponemos nada, y la red generará una melodía desde cero, basándose sólo en su entrenamiento.

El resultado es un archivo en formato MIDI. De esta manera, el usuario, músico o no, podrá guardar la melodía tal cual, o cambiar algunas notas y comenzar a trabajar desde esta primera idea.

La melodía será generada sobre un tiempo de 8 bars, porque en la música actual todos los temas son solamente un bucle repetido con diferentes variaciones de percusiones de melodía o de canto.

3.2 Herramientas

Usaremos el modelo MelodyRNN que está basado en una RNN usando neuronas con la LSTM que vimos anteriormente. Este modelo ya fue entrenado por Google con miles de archivos MIDI. Esta consideración es importante porque, si queremos obtener algunos resultados coherentes e interesantes, tendremos que usar un dataset lo más preciso y completo posible. Es decir, tomar archivos MIDI adaptados a un modelo de generación de melodía con una sola nota a la vez. Para entrenar el modelo, el dataset está extendido tomando los mismos archivos con las notas transpuestas en diferentes octavas, y luego se utiliza la técnica de *backpropagation.*

3.3 MelodyRNN

Este modelo está compuesto de una red LSTM de dos layers de 128 neuronas cada uno. Luego, usamos el modelo AttentionRNN, ya que pareció ser el de mejor calidad en algunos ejemplos de demostración en la documentación. El mismo utiliza el mecanismo *Attention* que vimos previamente**.** Los parámetros usados son los siguientes: *batch_size=128, 2 layers (128x128), dropout_rate=0,5, learning_rate =0,001, attention_length=40 (tamaño de la ventana de Attention).* Además, se usa un auto-codificador *KeyMelodyEncoderDecoder que* codifica los eventos repetidos, el tiempo y la clave de la melodía.

3.4 Experimentación

Hicimos varias pruebas de lo que describimos previamente para ver como MelodyRNN se comportaba. Los parámetros que quedaron constantes son el número de outputs (10), para obtener siempre melodías diferentes por cada generación que, si se solaparan una arriba de la otra, coincidirían. La duración de la melodía generada es de 8 bars también. De esta manera, el usuario podrá hacer un bucle con la misma duración, o guardar sólo 4 bars. Por defecto, hacemos una primera generación con *qpm=120*, *T=1.0* y un *input* de sólo una nota *C5* (o do).

Luego hacemos variar estos parámetros uno a la vez para ver como se comporta el modelo.

3.5 Resultados

Con la *temperatura* vimos que, si la bajamos, las melodías generadas son un poco monótonas hasta casi repetirse siempre las mismas cuando pusimos *T=0.5*. Aumentándola, podemos obtener algunos outputs más interesantes (hasta *T=1.25)* pero más desordenadas, hasta obtener algo totalmente desestructurado con *T=1.25*.

La modificación del *quarter per minutes* no influye mucho en los outputs, sino sólo en el número promedio de notas en la secuencia generada. Podemos arreglarlo en función del tipo de música que queramos hacer.

El *input* que ponemos en el modelo permite ubicar un intervalo de notas para la secuencia generada, y su primera nota determina también el tono de la gama que el modelo seguirá durante la generación.
Con un *input* MIDI de 4 bars, los 4 siguientes bars generados por el modelo quedan bien, pero es mejor aumentar un poco la temperatura, si no obtendremos la segunda parte de la melodía demasiado similar a la primera.

3.6 Observación

Podemos ver que todos las melodías generadas son en una escala musical en modo mayor, la más común. El tono seguido está definido con la primera nota del *input*.

Los archivos MIDI generados contienen algunas notas falsas de vez en cuando, pero el modelo persiste muy preciso.

Muchas veces, las 10 melodías generadas son diferentes, pero coinciden bastante bien.

Sin duda, con los parámetros por defecto, la generación es la mejor. Escuchando sus 10 outputs, y en modo de prueba práctica, imaginamos un instrumento y una función para cada secuencia MIDI generada. Luego, con una Digital Audio Workstation (DAW), creamos una pieza musical, agregando sólo percusiones simples y donde toda la parte melódica está generada con MelodyRNN.

4 Conclusión y futuro trabajo

En este trabajo, estudiamos y adaptamos un modelo de redes neuronales para que puedan servirle de asistencia a una persona durante su proceso de creación artística para crear una pieza musical. Vimos que el modelo MelodyRNN, que funciona gracias a una red RNN con neuronas LSTM y usando el mecanismo *Attention*, produjo algunos resultados bastante buenos aunque simples. Cuando los escuchamos suenan bien y hubieran podido venir de una persona real.

Para apoyar la validez de nuestros resultados, creamos un tema corto donde todas las notas son generadas por esta Inteligencia Artificial.

En un futuro trabajo, sería interesante volver a entrenar este modelo pero con archivos MIDI especificados en un genero de música especial, más complejo y moderno, o con el estilo de un artista, como lo hizo la empresa *Space150* con *TravisBott[8]*.

Más que reproducir, en este campo de investigaciones de Machine Learning aplicado a la música, el último paso sería crear nuevos géneros musicales, o al menos crear géneros híbridos, entrenando la red con archivos MIDI de dos tipos de música diferente.

Referencias

[1] Olah C., Team Google Brain, Understanding LSTM**,** *https://colah.github.io/posts/2015-08-Understanding-LSTMs/* , *2015*

[2] Problema de desvanecimiento de gradiente, *Wikipedia*

[3] Hochreiter, S. y Schmidhuber, J. Long short-term memory. *Neural computation*, 9(8):1735-1780, 1997.

[4] Briot J.P., Hadjeres, G., y Pachet, F. Deep Learning Techniques for Music Generation – A Survey. *ArXiv preprint arXiv:1709.01620v1*, 2017.

[5] Olah C., Carter S., Team Google Brain, Attention and Augmented Recurrent Neural Networks, *https://distill.pub/2016/augmented-rnns/*, *2016*

[6] Bahdanau D., Kyunghyun C., Bengio Y., Neural Machine Translation by Jointly Learning to Align and Translate, *arXiv:1409.0473, 2014*

[7] Waite E., Generating Long-Term Structure in Songs and Stories, **https://magenta.tensorflow.org/2016/07/15/lookback-rnn-attention-rnn**, 2016

[8] Space150, Travis Bott, **https://www.space150.com/work/v45-travisbott**, 2020

[9] Goolge Magenta, **https://magenta.tensorflow.org**