Generador de versos musicales en el idioma inglés por medio de procesamiento de lenguaje natural y redes neuronales

Espinosa de los Monteros Lechuga Jaime Daniel, Nava Romo Edgar Adrián, Salgado Gómez Alfredo Emilio
Kolesnikova Olga, López Rojas Ariel
Escuela Superior de Cómputo I.P.N. México D.F.
Tel. 57-29-6000 ext. 52000 y 52021.
E-mail: jamesdlechu@gmail.com, edgar.adrian97@gmail.com, alfredoe.sgomez@gmail.com

***Abstract*—The terminal work presented consists of the development of a web application, which allows users who access it, to generate lyrics of the pop genre through a model made by us, this was created using bidirectional LSTM neural networks.**

***Index Terms*—Bidirectional, LSTM, Music, Neural Networks.**

***Resumen*—El trabajo terminal presentado consiste en el desarrollo de una aplicación web, la cual permite a los usuarios que acceden a ella, generar letras musicales del género pop a través de un modelo generado por nosotros, este fue creado usando redes neuronales del tipo LSTM bidireccionales.**

***Palabras Clave*—Bidireccional, LSTM, Música, Redes neuronales.**

I. Introducción

En la actualidad la industria musical obtiene ganancias a través de la creación y divulgación de la música de manera física y digital [**actmusica**], dejando que aficionados y emprendedores de la música no tengan oportunidad de avanzar en su carrera, ya sea por la falta de creatividad, tiempo y/o recursos, haciendo que la creación de nuevas letras para sus canciones sea un gran obstáculo. Teniendo esto en cuenta, vimos una oportunidad y haciendo uso de las nuevas tecnologías, decidimos desarrollar una aplicación web la cual permite la generación de letras, las cuales pueden ser usadas posteriormente para integrarlas con sus temas musicales, creando canciones.

La generación de texto es una de las tareas más populares y desafiantes en el área de procesamiento del lenguaje natural. Recientemente, se han desarrollado una gran cantidad de trabajos (Generating Text with Recurrent Neural Networks [**refneuralnet**], Convolutional Neural Networks for Sentence Classification [**refneuralclass**]) los cuales propusieron generar texto utilizando redes neuronales recurrentes y redes neuronales convolucionales. Sin embargo, la mayoría de los trabajos actuales solo se enfocan en generar una o varias oraciones, ni siquiera un párrafo largo y mucho menos una canción completa.

Las letras de canciones, como un tipo de texto, tienen algunas características propias, estas se constituyen de coros, estribillos, rimas, versos y en algunos casos, patrones de repetición. Se entiende por coro que se trata de una estrofa la cual se repite varias veces dentro de una composición, acentuando la idea más importante de la canción, el estribillo es un conjunto de palabras o frases con la que se inicia la composición y la cual se repite al final de una estrofa de esta. Mientras que el verso es la parte encargada de comenzar a desarrollar la idea a transmitir, trata de contarnos el tema de la canción. Estas características hacen que generar letras musicales sea mucho más difícil que generar textos normales.

La generación de letras por medio de inteligencia artificial dado un estilo y un tema es un asunto poco trabajado. Por lo tanto, nos enfocamos en trabajar este problema. Estamos interesados en ver si el modelo propuesto puede aprender diferentes características en un género musical y generar letras que sean acorde a este.

II. Metodología

Se entiende por Inteligencia Artificial a una simulación de procesos computacionales cognitivos para que simulen el comportamiento de una mente humana, estos comportamientos abarcan diferentes áreas de investigación, como el aprendizaje, comunicación, percepción, razonamiento y la capacidad de desarrollarse en entornos más complejos.

A. Red neuronal

Una red neuronal es una herramienta derivada de la inteligencia artificial que utiliza modelos matemáticos para ser utilizada como un mecanismo de predicción del texto. Son una forma de hacer que las computadoras aprendan, donde la computadora aprende a realizar alguna tarea analizando ejemplos de entrenamiento. Por lo general, estos ejemplos han sido etiquetados previamente.

Las redes neuronales son modeladas vagamente asemejando el cerebro humano, esta consiste en miles de millones de nodos que realizan el procesamiento de la información los cuales están densamente interconectados.

La complejidad de las neuronas reales se abstrae mucho cuando se realiza el modelado de neuronas artificiales. Estas consisten básicamente en entradas, que se multiplican por pesos (fuerza de las respectivas señales), y luego calculada

por una función matemática que determina la activación de la neurona. Otra función (que puede ser la de identidad) calcula la salida de la neurona artificial (a veces en dependencia de un cierto umbral). Las redes neuronales combinan neuronas artificiales para procesar información.[**neuronart**]

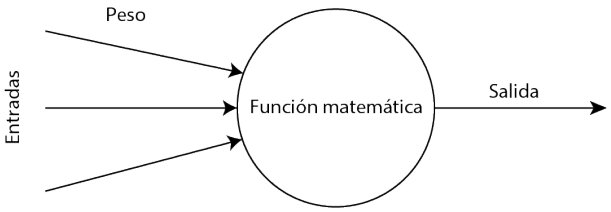

Fig. 1. Neurona artificial.

B. Recurrent Neural Network (RNN) – Long Short Term Memory (LSTM)

Una red neuronal recurrente es un tipo de red neuronal artificial donde la salida de alguna capa en particular es salvada y sirve para retroalimentar la entrada de esta capa, lo cual ayuda a predecir futuras salidas de esta. La primera

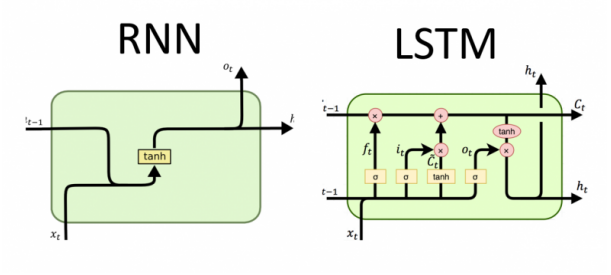

Fig. 2. Diagrama de las redes neuronales RNN y LSTM.

capa está formada de la misma manera que la Feed-forward Neural Network, es decir, solo pasa la información que entra a la siguiente capa inmediata, posteriormente la siguiente capa con el paso del tiempo comenzará a retroalimentarse, pero manteniendo la propagación frontal. Haciendo uso de esta retroalimentación la capa en futuras operaciones puede realizar predicciones, si estas predicciones no son los resultados esperados, el sistema aprende y trabaja para corregir sus futuras predicciones.[**redneurnn**]

Este tipo de redes neuronales se distinguen por su "memoria", ya que toman información de entradas anteriores para influir en la entrada y salida actuales.

C. Obtención del conjunto de datos

El primer paso para poder empezar a trabajar con nuestro modelo fue obtener un conjunto de datos que cumpliera con las características que se necesitaban, por eso nos dimos a la tarea de buscar uno. En Kaggle [**kaggle1**] se encontró un dataset llamado "Song lyrics for 6 musical genres" [**kaggleDataset**] el cual contenía los datos necesarios y contaba con un total de 160,790 letras, las cuales pertenecían a 6 tipos de géneros musicales:

- Rock
- Pop
- Hip Hop
- Samba
- Sertanejo
- Funk Carioca

Debido a que estas letras de canciones pertenecían a distintos géneros e idiomas se procedió a realizar una limpieza a la base de datos para quedarnos únicamente con las letras de canciones pertenecientes al genero pop y que estuvieran en el idioma inglés, además haciendo uso de la librería de "Pandas" y siguiendo el estándar de limpieza de datos: [**data˙cleaning**]

1) Remover caracteres innecesarios
2) Eliminar Duplicados
3) Evitar errores ortográficos de similitud
4) Convertir los tipos de dato
5) Tratar los valores nulos o faltantes

Se obtuvo como resultado un conjunto de datos útil, el cual solo cuenta con los datos necesarios para después poder tratarlos y utilizarlos para empezar el entrenamiento del modelo de la red neuronal, cabe destacar que este proceso es necesario hacerlo solo una vez para cada género musical ingresado, en caso de que se lleguen a necesitar más datos, será necesario repetir el proceso.

D. Modelo

Para el desarrollo del modelo se utilizó una libreta de Kaggle. Cabe mencionar que el lenguaje de programación utilizado para el desarrollo del modelo fue Python, por ende, lo primero que se trabajó en la libreta, fue realizar la importación de librerías que se fueran a necesitar, en este caso las más importantes son:

- Pandas: la cual nos permite manipular y analizar la información.
- Wordcloud: es una herramienta que nos ayuda al momento de generar la siguiente palabra de una oración.
- Tensorflow: una herramienta que nos ayuda en el procesamiento del aprendizaje automático.
- Keras: una herramienta que nos ayuda en el procesamiento del aprendizaje profundo.

A continuación, procedemos a importar el conjunto de datos previamente trabajado y limpiado, pero en esta ocasión apoyándonos de la librería de pandas, solo nos vamos a quedar con letra de estas canciones, ya que es la información que nos interesa trabajar y la cual vamos a estar manipulando.

Al usar Kaggle debemos tener en cuenta sus limitantes, en este caso se cuenta solamente con 16Gb de RAM, un disco de 73Gb y una GPU de 13GB para el procesamiento de nuestro conjunto de datos, por eso debemos limitar nuestro modelo, en lugar de utilizar todos los datos (28441), vamos a trabajar solo con 700 letras de canciones por el momento.

Utilizando estas 700 lyrics, se decidió obtener información de ellas, siendo más específicos, estadísticas sobre el número de palabras en cada canción, esto con el fin de determinar la frecuencia de distribución del número de palabras de cada texto y para tener una idea del promedio de palabras, esto con el fin de tenerlo en cuenta al momento de realizar la generación de texto. Lo que podemos observar en la imagen

count	700.000000
mean	248.065714
std	97.323252
min	31.000000
25%	178.000000
50%	233.000000
75%	309.000000
max	589.000000

Fig. 3. Estadísticas de las palabras

anterior es:

- Count: el cual es el número de canciones analizadas.
- Mean: el promedio de palabras por canción.
- Std: la desviación estándar de las palabras.
- Min: la menor cantidad de palabras encontradas en una canción.
- Max: la mayor cantidad de palabras encontradas en una canción.

Además, se le realizo una tokenizacion a las letras de nuestras 700 canciones, esto quiere decir que se separó cada palabra y cada palabra se convirtió en un número. Para este proceso se hizo uso de Keras y la clase Tokenizer(). Antes de la

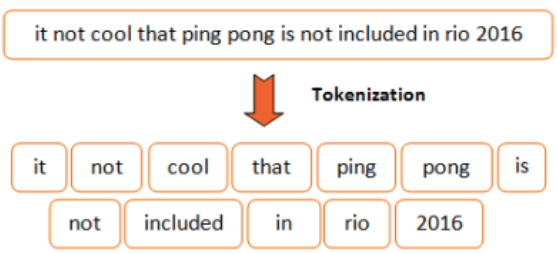

Fig. 4. Tokenizado de las palabras [**tokenimagen**]

generación del modelo, es necesario normalizar todas las oraciones a una misma longitud estándar, para evitar el desbordamiento de la memoria y conseguir que las capas del modelo sean mucho más profundas, este es un proceso simple el cual consta de agregar ceros al comienzo del texto, dando como resultado capas del mismo tamaño, a este proceso se le conoce como padding.

La posición donde se sumarán los ceros viene determinada por el relleno del argumento, en este caso, se hará al comienzo de la secuencia.

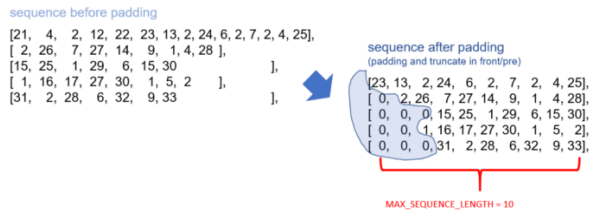

Fig. 5. Padding del tokenizado de las palabras

E. Creación del modelo

En este caso se utilizó el modelo LSTM Bidireccional, este tipo de redes neuronales se ejecutan como su nombre lo indica: en dos direcciones. Esto quiere decir que va del pasado al futuro y viceversa, así es como el modelo conserva la información de ambos estados en cualquier momento. Las redes neuronales de LSTM se utilizan principalmente cuando el contexto está involucrado.

Los modelos en Keras se definen como una secuencia de capas, y el modelo secuencial se trata de agregar capas de una en una. Las capas son el componente básico de la red neuronal.

Dentro de estas capas podemos encontrar:

e.1) Embedding o incrustación

Es una capa central, solo se puede usar como la primera capa en un modelo, convierte los números enteros positivos en vectores densos de un tamaño fijo (el primer parámetro es el tamaño del vocabulario, el segundo parámetro es la dimensión de la incrustación densa y el tercer parámetro es sobre la longitud de las secuencias, este se requiere ya que usaremos una capa densa más adelante)

e.2) Bidireccional

Es una capa recurrente, una envoltura bidireccional para las redes neuronales de tipo RNN's que recibirá una capa como entrada, siendo la capa LSTM la que elegimos, recibirá un entero positivo como entrada que se refiere a la cantidad de nodos de salida que se deben devolver.

e.3) Dropout

Es una capa de regularización. Esta capa establece aleatoriamente las unidades de entrada en 0, con una frecuencia del valor que le pasamos, en cada paso durante el tiempo de entrenamiento, lo que ayuda a evitar el sobreajuste.

e.4) Densidad

Es una capa central y una capa de red neuronal densamente conectada. Recibe como primer parámetro un entero positivo que se refiere a la cantidad de nodos de salida que deben devolverse. El segundo parámetro es el llamado activación que define el tipo de predicciones que puede hacer el modelo; para el tipo de problema que estamos considerando, el que se adapta mejor es softmax, que genera un vector de valores (entrada) que puede interpretarse como probabilidades de ser utilizado.

e.5) Método de compilación, algoritmo de optimización y métrica de rendimiento

Perdida: También conocida como función de costos; funciona durante el proceso de optimización y su función es calcular el error del modelo. La entropía cruzada se utiliza para estimar la diferencia entre una distribución de probabilidad estimada y predicha. Se utilizará categorical_cross-entropy porque es más adecuado para este tipo de problemas y se usa casi universalmente para entrenar redes neuronales de aprendizaje profundo debido a los resultados que produce.

Optimización: Se encarga de reducir las pérdidas y brindar los resultados más precisos posibles. Adam es la opción elegida porque es la mejor opción que ofrece Keras para entrenar la red neuronal en menos tiempo y de manera más eficiente. Earlystop detendrá el entrenamiento si el modelo ha dejado de mejorar, esto se verificará al final de cada epoch. En este caso, la ”precisión”o “accuracy” se utilizará como métrica de rendimiento. El método fit es el encargado de entrenar el modelo para el número fijo de epochs dados.

Quedando como resultado el código del modelo de la siguiente forma:

```
model = Sequential()
model.add(Embedding(total_words, 40, input_length=
    max_sequence_len-1))
model.add(Bidirectional(LSTM(250)))
model.add(Dropout(0.1))
model.add(Dense(total_words, activation='softmax'))
model.compile(loss='categorical_crossentropy', optimizer='
    adam', metrics=['accuracy'])
earlystop = EarlyStopping(monitor='loss', min_delta=0,
    patience=3, verbose=0, mode='auto')
history = model.fit(X, y, epochs=20, verbose=1, callbacks=[
    earlystop])
```

e.6) Importación del modelo y función de generación

Una vez completado el entrenamiento de nuestro modelo, lo que falta es importarlo para probar cómo funciona, en nuestro caso nombramos al modelo “song_lyrics_generator” y se importó de la siguiente forma:

```
from keras.models import load_model
model = load_model('../input/songlyricmodel/
    song_lyrics_generator.h5')
```

Ya que contemos con el modelo importado, se creó una función que se utilizará para generar la letra de una canción utilizando el modelo previamente entrenado, predecirá las siguientes palabras en base a la palabra(s) de entrada suministradas como 'seed_text'. Para que esto funcione, se debe aplicar una tokenización al seed_text, luego se aplicará un relleno a las secuencias generadas y se pasará al modelo entrenado para que se pueda predecir la siguiente palabra.

```
def complete_this_song(seed_text, next_words):
    for _ in range(next_words):
        token_list = tokenizer.texts_to_sequences([seed_text
    ])[0]
        token_list = pad_sequences([token_list], maxlen=
    max_sequence_len-1, padding='pre')
        predicted = model.predict_classes(token_list,
    verbose=0)
        output_word = ""
        for word, index in tokenizer.word_index.items():
            if index == predicted:
                output_word = word
                break
        seed_text += " " + output_word
    return seed_text
```

III. Resultados

El modelo generado y entrenado dentro de la plataforma de Kaggle se extrajo y se llevó completamente a la práctica, funcionando dentro de una aplicación web, permitiendo a aquellos usuarios que ingresaran a la siguiente dirección (x) pudieran probar de primera mano la generación de texto realizada por nuestro modelo, obteniendo letras de canciones a partir de una o varias palabras en idioma inglés introducidas por el usuario.

Destacamos que la generación de texto de nuestro modelo no es el mejor, esto debido a las limitantes con las que contábamos al momento de realizar el entrenamiento del modelo, ya que este se fue realizando en bloques, es decir de 700 en 700 letras de canciones en lugar de darle a procesar todos los textos. Pero a pesar de este inconveniente, consideramos que da buenos resultados.

A continuación, dejamos algunos ejemplos de los textos obtenidos:

IV. Conclusiones

Hoy en día el desarrollar aplicaciones y/o herramientas que hacen uso de algún tipo de inteligencia artificial, redes neuronales o procesamiento del lenguaje natural es ingresar a una de las áreas de estudio mas amplia y compleja que en los últimos años ha tenido un gran desarrollo, así como un gran apoyo por empresas privadas y gobiernos.

Con este proyecto de trabajo terminal se intentó dar solución al problema, dando a estudiantes o aficionados de la música una herramienta que les de un texto a partir de una idea que ellos propongan y usarlo sin realizar algún cambio a este texto generado o adaptándolo a las necesidades y gustos de cada artista.

La aplicación web fue desarrollada de tal forma que pueda ser accesada desde cualquier dispositivo móvil que cuente con acceso a internet y un navegador web, así como se hizo amigable para el usuario invitándolo desde que ingresa a generar su propia letra de canción.