Generación De Texto Con Redes Neuronales Recurrentes

Yana Alanoca Cesar Florencio
Universidad Nacional de Ingenieria
Lima, Perú
cesar.yana.a@uni.pe

Rivera Granados Franklin Félix
Universidad Nacional de Ingeniería
Lima, Perú
friverag@uni.pe

***Resumen*—En este proyecto, nos sumergimos en el mundo de las Redes Neuronales Recurrentes, donde sabemos que existen muchos tipos de arquitecturas. Para esta investigación, nos centraremos en la arquitectura de Redes Neuronales Recurrentes LSTM para la generación de texto basado en Andrej Karpathy´s [15]. Al principio se describen distintos términos que nos ayudan a entender mejor el funcionamiento de dicha Red Neuronal Recurrente.**

Se establece una metodología de desarrollo que consiste en un proceso de Tokenización, método de procesado de la data,creación de mini-batches, entrenamiento y proceso de predicciones del siguiente carácter a generar.

Se realizaron distintos experimentos variando el número de batches(1 y 8), en ambos casos usando la misma longitud de texto se obtuvo resultados similares o iguales, pero al momento de variar la longitud del texto al 10 % de la anterior, el texto generado fue erróneo, dejando la puerta abierta para mejorar la precisión de dicha Red Neuronal Recurrente.

Palabra claves — RNN, RN,LSTM.

I. Introducción

Las Redes Neuronales Recurrentes son una de las principales arquitecturas del Machine Learning, muy usadas por ejemplo en los sistemas de reconocimiento de voz o en el análisis de video, o en el procesamiento del lenguaje natural.

Las Redes Neuronales [9] o las Redes Convolucionales [10] permiten procesar un solo dato a la vez (como un sonido o una imagen), pero si tenemos una secuencia de sonidos (una conversación) o de imágenes (un video), este tipo de arquitecturas no estarán en capacidad de procesar este tipo de secuencias.

Las Redes Neuronales Recurrentes resuelven este inconveniente, pues son capaces de procesar diferentes tipos de secuencias (como videos, conversaciones, texto) y, a diferencia de las redes neuronales o convolucionales, no se utilizan para clasificar un dato en particular sino que también están en la capacidad de generar nuevas secuencias [11].

II. Objetivos del estudio

Comprender el funcionamiento de las redes neuronalas recurrentes y como estas nos permiten generar texto.

II-A. Objetivos específicos

1. Diseñar una Red Neuronal Recurrente LSTM en Pytorch para crear un modelo que permita generar texto.
2. Evaluar la efectividad de nuestra Red Neuronal Recurrente LSTM que genera texto.

III. Fundamento Teórico

III-A. Redes Neuronales Recurrentes

En primer lugar definamos a qué nos referimos con **instante de tiempo.** Para una secuencia, el instante de tiempo es simplemente un número entero que define la posición de cada elemento dentro de la secuencia.

Así por ejemplo, en la palabra ”r-e-c-u-r-r-e-n-t-e”, el instante de tiempo 1 correspondiente al primer carácter de la secuencia(la letra ”r”), el instante de tiempo 2 al segundo carácter (la letra ”e”) y así sucesivamente.

Nos referimos a x_t como la entrada a la red recurrente en el instante de tiempo t, y a y_t como la salida en el instante de tiempo t [12].

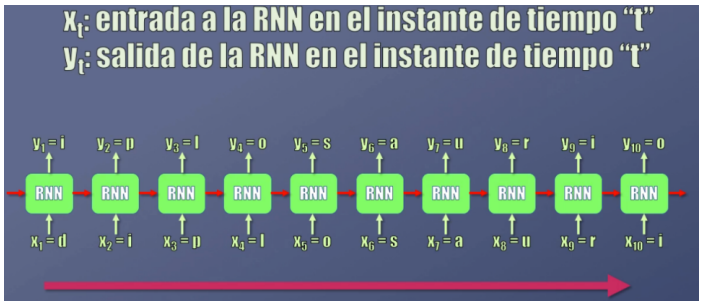

Figura 1. Definición de la entrada (x) y la salida (y) en cada instante de tiempo [12]

III-A1. La Red Recurrente: entradas y salidas: Pero, ¿Cómo logra la Red Recurrente predecir correctamente el siguiente carácter en la secuencia? Es decir, ¿dónde está la memoria que mencionamos anteriormente?

La respuesta está precisamente en un elemento importante que observamos en la Figura 1: las flechas horizontales de color rojo. Se oberva que en cada instante de tiempo la red tiene realmente dos entrada y dos salidad.

Las entradas son el dato actual (x_t) y la activación anterior (a_{t-1}) mientras que las salidas son la predicción actual (y_t) y la activación actual (a_t). Esta activación también recibe el nombre de "hidden state.º estado oculto:

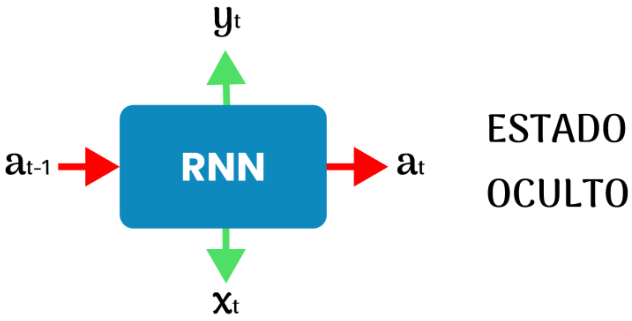

Figura 2. Las entradas y salidas de una Red Recurrente [12]

Son estas las activaciones las que corresponden precisamente a la memoria de la red, pues permiten preservar y compartir la información entre un instante de tiempo y otro [12].

Veamos cómo se genera la predicción y la activación, y cómo estas se relacionan con la memoria de la red.

Para calcular la salida y la activación de la Red Recurrente, a partir de sus dos entradas, se usa la misma lógica de una Neurona Artificial convencional.

En decir una Red Neuronal Recurrente tiene una entrada (x) y genera una salida (y), y que la salida es el resultado de aplicar dos operaciones al dato de entrada: una transformación y una función de activación no-lineal.

Principio de funcionamiento de una Neurona Artificial:

$$X \Rightarrow neurona\_artificial \Rightarrow y = f(WX + b)$$

Donde:

Función convencional: f

Transformación: $WX + b$

En el caso de las Redes Recurrentes, la activación se calcula de manera similar, y es el resultado primero de transformar los datos de entrada (es decir la activación anterior y la entrada actual) y luego llevarlos a una función de activación no-lineal.

Obtención de la activación en una Red Neuronal Recurrente:

$$a_{t-1} \Rightarrow RNN \Rightarrow a_t = f(W_{aa}a_{t-1} + W_{aX}X_t + b_a)$$

Donde:

Función convencional: f

Transformación: $W_{aa}a_{t-1} + W_{aX}X_t + b_a$

III-B. *Modelos de lenguaje a nivel de personaje*

Consiste en entrenar a una RNN, donde le daremos una gran cantidad de texto, a esta RNN, con el fin que pueda modelar la distribución de la probabilidad del siguiente carácter en la secuencia, logrando generar un nuevo texto. Como ejemplo básico, le daremos un conjunto de datos, "hello", donde nuestra RNN, tendrá que predecir el siguiente carácter, veamos.

Se tendrá una fuente de 4 ejemplos de entrenamiento separados: 1. La probabilidad de 'e' probablemente debería estar dado el contexto de 'h', 2. 'l' debería estar probablemente en el contexto de 'he', 3 . 'l' probablemente también debería ser dado el contexto de 'hel', y finalmente 4. 'o' probablemente debería ser dado el contexto de "hell" [13].

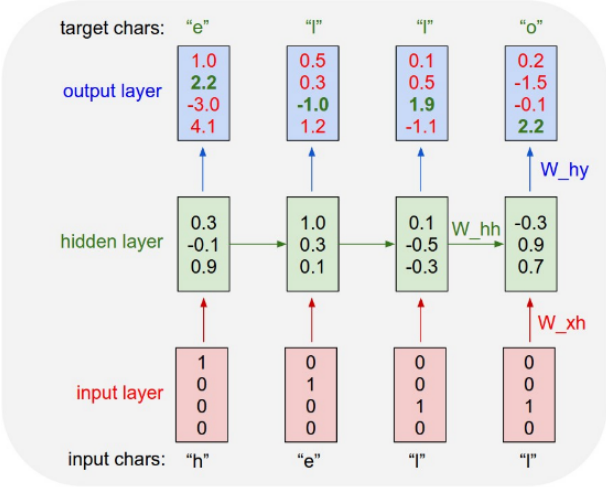

Figura 3. ejemplo de RNN con capas de entrada y salida de 4 dimensiones y una capa oculta de 3 unidades (neuronas) [13]

En la anterior imagen, vemos un diagrama, donde se muestra las activaciones en el pase hacia adelante, en el caso que la RNN, se alimentara de los carácteres "hell". En el otro caso, que es el de salida, esta contiene confedencias en la RNN,la misma que se le asigna para el siguiente carácter.

III-C. *Long short-term memory(LSTM)*

Las redes LSTM, son un tipo de arquitectura de las redes neuronales recurrentes, más utilizadas en la actualidad. Este tipo de arquitecturas, son capaces de recordar datos relevantes en la secuencia, y de preservarlo en uno o varios instantes de tiempo, teniendo una memoria a corto plazo. Su forma de ejecutarse es tomar una decisión sin enfocarse en la totalidad del texto, sino de enfocarse, en solo las palabras relevantes, omitiendo lo demás [5].

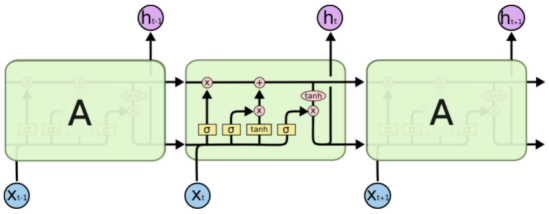

Figura 4. El módulo de repetición en un LSTM contiene cuatro capas que interactúan [14]

IV. Estado Del Arte

IV-A. Redes Neuronales Recurrentes Para El Análisis De Secuencias

En este artículo con el mismo nombre Redes Neuronales Recurrentes Para El Análisis De Secuencias [1] se estudia de la importancia de las redes neuronales recurrentes y de su naturaleza en análisis de secuencias o señales donde es muy importante tener en cuenta el pasado o el futuro.

Se muestra la fortaleza de estos métodos para analizar secuencias de tamao variable, por su despliegue en el tiempo en función del tamaño de la entrada. Se construyen específicamente redes neuronales recurrentes bidireccionales, como una especificación de redes recurrentes, mostrando la potencialidad de las mismas en sistemas no causales, donde las entradas pueden depender de entradas de tiempos pasados y futuros. Además se desarrolla Una plataforma para implementar redes recurrentes dinámicas, con algoritmo de aprendizaje Backpropagation Trough Time; que permiten desarrollar redes para cualquier problema donde las entradas son secuencias analizadas en el tiempo y la salida son otras secuencias o simplemente descriptores de funciones o propiedades de las mismas.

IV-B. Modelos De Redes Neuronales Recurrentes En Clasificación De Patentes

En este artículo del mismo nombre Modelos De Redes Neuronales Recurrentes En Clasificación De Patentes [2] se centra en la clasificación automática de patentes usando redes LSTM (Long-Short Term Memory). Para empezar se describen distintos métodos de tratamiento y extracción de características de textos y de clasificación por aprendizaje automático.

Se establece una metodología de desarrollo y pruebas de modelos para poder evaluar con facilidad los distintos experimentos. Dentro de esta metodología se incluye un framework de flujo de datos y entrenamiento de modelos así como pequeñas utilidades para poder replicar todos los experimentos en entornos virtualizados con docker pero con acceso a GPUs. Además, este framework realiza un guardado automático de los resultados intermedios de los distintos módulos de procesado de texto (que también ha habido que implementar) para así conseguir que operaciones costosas previas al entrenamiento se ejecuten solo cuando sea necesario y no más de una vez.

Se realizaron unos experimentos con distintas variaciones de modelos con LSTM, algunos con un mapeo precalculado y otros entrenando un mapeo de cero. La gran mayoría fueron poco conclusivos y no tuvieron resultados muy buenos, pero dejan abierta la puerta a sencillas mejoras e indican los pasos a seguir para mejorar la precisión.

IV-C. Redes Neuronales Convolucionales y Redes Neuronales Recurrentes

En este artículo de nombre Redes neuronales convolucionales y redes neuronales recurrentes en la transcripción automática [3] trata el problema de que las transcripciones varian de persona en persona, dependiendo de su experiencia y habilidad, siendo una más acertada uno que otros. La forma en que se ha buscado solucion a este problema es con el uso de redes neuronales, específicamente redes neuronales convolucionales y redes neuronales recurrentes. Con el uso de ambos tipos de redes neuronales se propone en este artículo una metodología mixta que permita automatizar el proceso de transcripcion de una lengua.

En esta investigación se encontró que la forma en que se pueden combinar estas dos arquitecturas es dejando a las redes neuronales convolucionales extraer características acusticas de alto nivel, mientras que a las redes neuronales recurrentes se les asigna la tarea de clasificar secuencias. Así se reducira la posibilidad de error en las transcripciones eliminando el factor subjetivo que subyace al trabajo hecho por humanos.

V. Metodología

Detallaremos las herramientas y metodologías que se usarán para el desarrollo de nuestra Red Neuronal Recurrente LSTM para generar texto:

V-A. Numpy

Es una libreria de python [4], que permite la creación de cálculos, usada frecuentemente en operaciones con matrices el cual nos será muy útil para el desarrollo de nuestra RNN.

V-B. PyTorch

PyTorch [6] es marco de aprendizaje automático (ML) de código abierto basado en el lenguaje de programación Python y la biblioteca Torch. Es una de las plataformas preferidas para la investigación de aprendizaje profundo . El marco está diseñado para acelerar el proceso entre la creación de prototipos de investigación y la implementación.

V-C. Métricas

V-C1. Tokenización: En este apartado crearemos dos diccionarios para convertir los carácteres en enteros, al hacer esto será más fácil usarlo para la entrada de nuestra red neuronal.

V-D. Procesar la Data

Las redes neuronales solo aceptan números, entonces vamos a tener que hacer que nuestros datos sean números y esto podemos hacerlo con **one-hot encode** que significa que cada carácter es convertido a un entero y luego en una columna de vectores que para el entero que corresponde tendrá un 1 y el resto lleno de 0.

V-E. Haciendo mini-batches para entrenamiento

Para entrenar la data, vamos a crear mini-batches y con esto hacer el entrenamiento. Recordemos que nuestro batch debe tener múlipltes secuencias de una cantidad de números. Este fuera un ejemplo:

Secuencia de inicio:
[1 2 3 4 5 6 7 8 9 10 11 12]

Batch size = 2:
[1 2 3 4 5 6]
[7 8 9 10 11 12]

Longitud de la secuencia:
[1 2 3]
[7 8 9]

Creando Batches

- Primero vamos a tener que descartar algo de texto, para tener un batch completo.
- Luego vamos a dividir nuestro arreglo en N batches.
- por último vamos a tener nuestro arreglo y podremos iterar por nuestros mini-batches.

V-F. Entrenamiento

- Usaremos **Adam optimizer** y **Cross-Entropy loss** ya que se hace una clasificación de carácteres.
- Usaremos *clip_grad_norm_* para evitar la explosión del gradiente.

V-G. Haciendo predicciones

Una vez que el modelo está entrenado, querremos probarlo y hacer predicciones sobre los próximos personajes. Para muestrear, pasamos un carácter y hacemos que la red prediga el siguiente carácter. Luego tomamos ese personaje, lo devolvemos y obtenemos otro personaje predicho.

La salida de nuestro RNN es de una capa completamente conectada y genera una **distribution of next-character scores**.

Para obtener el siguiente carácter, aplicamos una función **softmax**, que nos da una distribución de probabilidad que luego podemos muestrear para predecir el siguiente carácter.

Nuestro modelo usa una **softmax**, pero podemos añadir algo de aleatoridad para escoger uno de los carácteres más probables con **Top K** [7], asi puede generar un texto algo aleatorio y aveces absurdo.

El **Top K** nos devuelve una tupla con nombre de (valores,índices), donde los índices son los índices de los elementos en el tensor de entrada original.

VI. Experimentación y resultados

La experimentación se ejecutó en Google Colab en un entorno de GPU.

VI-A. Conjunto de datos

Para probar el funcionamiento del modelo basta con tener un archivo .txt, donde no es de especial interés el contenido de dicho archivo y si la longitud del texto dentro del archivo que es de 691699.

El entrenamiento se realizó primero con 1 batch y luego con 8 batches, se obtuvo los siguientes resultados:

1. Para 1 batch

   El texto inicial que pasamos fue **El libro empieza años** y el texto generado fue lo que se oberva en la Figura 5.

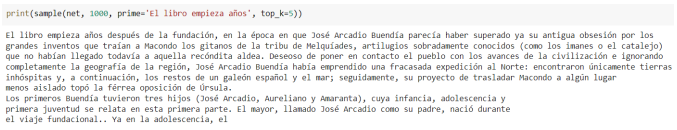

Figura 5. Texto generado con 1 batch

2. Para 8 batches

   El texto inicial que pasamos fue **El libro empieza años** y el texto generado fue lo que se oberva en la Figura 6.

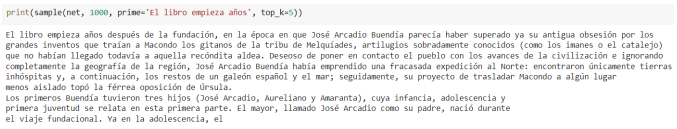

Figura 6. Texto generado con 8 batches

En este mismo apartado haremos una prueba con otro archivo que contiene un texto de longitud 60489 que es aproximadamente el 10 % de la longitud del anterior archivo y obtenemos el resultado mostrado en la Figura 7.

Figura 7. Texto generado con 8 batches

VII. Discusión De Resultados

De la Figura 5 y 6 podemos observar que el texto generado son similares a pesar que el número de batches son distinos, 1 y 8 respectivamente.

Pero si observamos la Figura 6 y 7 el texto generado en ambos casos es distinto, dado que en la Figura 6 se generó el texto correctamente y en la Figura 7 el texto generado es incorrecto. Entonces la longitud del texto influye en el proceso de entrenamiento, es decir, si la longitud del texto es grande el texto se generará de manera similar o igual al texto original y en caso contrario el texto se generará de manera errónea.

VIII. Conclusiones

Pudimos comprender como funciona las Redes Neuronales Recurrentes y como nos permiten generar texto.

- Se logró diseñar la Red Neuronal Recurrente LSTM en Pytorch para poder generar texto.
- Se concluye que el modelo diseñado es efectivo si la longitud del texto es grande, en caso contrario el texto se genera de manera errónea o poco similar al texto original.

Referencias

[1] Cruz, I. B., Martínez, S. S., Abed, A. R., Ábalo, R. G., Lorenzo, M. M. G. (2007). Redes neuronales recurrentes para el análisis de secuencias. Revista Cubana de Ciencias Informáticas, 1(4), 48-57.

[2] Guridi Mateos, Guillermo. Modelos de redes neuronales recurrentes en clasificación de patentes. BS thesis. 2017.

[3] Prieto, Juan Sebastian Gelvez. Redes neuronales convolucionales y redes neuronales recurrentes en la transcripción automática."

[4] Numpy: https://aprendeconalf.es/docencia/python/manual/numpy/

[5] LSMT: https://www.codificandobits.com/blog/redes-lstm/

[6] PyTorch: https://searchenterpriseai.techtarget.com/definition/PyTorch

[7] Top K: https://pytorch.org/docs/stable/torch.htmltorch.topk

[8] https://karpathy.github.io/2015/05/21/rnn-effectiveness/

[9] Redes Neuronales: https://www.codificandobits.com/blog/que-es-una-red-neuronal/

[10] Redes Convolucionales: https://www.codificandobits.com/blog/redes-convolucionales-introduccion/

[11] Redes Neuronales Recurrentes: https://www.codificandobits.com/blog/introduccion-redes-neuronales-recurrentes/introducci%C3%B3n

[12] Redes Neuronales Recurrentes: https://www.codificandobits.com/blog/redes-neuronales-recurrentes-explicacion-detallada/

[13] La efectividad irrazonable de las redes neuronales recurrentes: https://karpathy.github.io/2015/05/21/rnn-effectiveness/

[14] https://colah.github.io/posts/2015-08-Understanding-LSTMs/

[15] Github del código usado como referencia: https://github.com/karpathy/char-rnn