La efectividad de los modelos de lenguaje a nivel carácter y por qué las RNN siguen siendo geniales

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*—**

Índice de Términos— RNN, RN,LSTM.

I. Introducción

Desde la creación de la primera computadora, el hombre ha ido mejorando más y más la funcionalidad de las computadoras, hoy en día, podemos apreciar súper computadoras que procesan grandes cantidades de datos en un instante, a diferencia de las computadoras antiguas, que se demoran horas y hasta días. Sin embargo, y pese a tener computadoras con grandes potenciales, el hombre ha ido en la búsqueda de crear la autonomía de una computadora, es decir darle inteligencia propia, dándole a la computadora, el don de poder elegir una respuesta optima, similar a la toma de decisiones del ser humano. A esta tecnología se le llama Inteligencia Artificial, donde las maquinas, tienen la capacidad de poder, tomar decisiones óptimas. Se ha creado múltiples arquitecturas de esta tecnología, pero si buscamos la más eficaz, seria hablar de Redes Neuronales, en donde esta arquitectura se comporta como una inteligencia autónoma, pero que en realidad se basa en la comparación de respuestas guardadas de otros análisis. En este trabajo, nos enfocaremos en un tipo de redes neuronales, que son las redes neuronales recurrentes, que a diferencia de la anterior, analizan dato por dato, las RNN, analizan una secuencia de datos, abriendo un abanico de múltiples funciones que se le pueden dar a esta red neuronal recurrente, como por ejemplo, la detección de textos, la detección de imágenes(es decir, videos), la detección de sonidos (música), etc. Sin embargo, hablar de Redes Neuronales Recurrentes, es de hablar de temas complejos, ya que no solo es crear una RNN, sino también, se debe de entrenarla, para que esta RNN, pueda predecir otras respuestas futuras, es decir, aprender.

El objetivo de nuestro proyecto es desarrollar una red neuronal recurrente LSTM a nivel de carácter en PyTorch, el modelo entrenará con un texto, donde se analizará carácter por carácter para generar un nuevo texto, mediante la predicción probabilística. La red a desarrollar está basada en Andrej Karpathy's RNNs [11] e implementada en Python.

En este trabajo de investigación, podremos apreciar en la sección 1, una breve introducción, donde daremos una breve historia que apunta a conocer lo general hacia lo específico, que son las RNN, tema que nos interesa. En la segunda parte, sección 2, agregaremos teorías que nos ayudara a entender algunas palabras técnicas que existan en la metodología. En la sección 3, también veremos teoría, pero, no son temas generales, sino temas en que nos apoyamos para poder construir nuestra RNN, cabe resaltar que están basadas en artículos científicos. En la sección 4, comenzaremos con construcción de la red neuronal recurrente, donde detallaremos como se construyó la RNN desde 0, hasta su construcción total, también veremos el entrenamiento de esta RNN. En la sección 5, llevaremos nuestra RNN a la práctica. En la sección 6, compararemos los resultados que nos arroja por experiencia nuestra, con la experiencia de otros investigadores, dándole una explicación, del por qué los resultados salieron no fueron óptimos o fueron diferentes, si es q el caso fuera, erróneo. En la seccion 7, mencionares las conclusiones que nos dejó el trabajo de investigación, y en la seccion 8, veremos las citas, de donde nos apoyamos, para poder investigar.

II. Fundamento Teórico

II-A. Redes Neuronales Recurrentes

Las redes neuronales recurrentes o RNN, se construyeron con el fin, de darle una solución a los análisis de secuencia, como por ejemplo, el análisis de textos enteros, videos (muchas imágenes), audios, etc. Esto ocurrió por que la red neuronal convencional, no podía analizar muchas imágenes a la vez, pero si una por una, limitando, su funcionalidad.

Desde la creación de las RNN, se han visto que son muy difíciles de entrenar, debido a las necesidades de computación, es por ello que no se han vuelto a ver.

La gran diferencia entre una RNN con una red neuronal convencional, es que la red neuronal convencional,

solo mira hacia adelante, es decir que no recuerda los valores anteriores, sin embargo la RNN, tiene la posibilidad de ver hacia atrás, una especie de retroalimentaciones entre las neuronas dentro de las capas.

Imaginemos la RNN más simple posible, compuesta por una sola neurona que recibe una entrada, produciendo una salida, y enviando esa salida a sí misma, como se muestra en la siguiente figura:

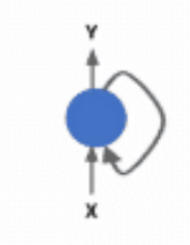

Figura 1. Una sola neurona

En cada instante de tiempo (también llamado timestep en este contexto), esta neurona recurrente recibe la entrada xde la capa anterior, así como su propia salida del instante de tiempo anterior para generar su salida y. Podemos representar visualmente esta pequeña red desplegada en el eje del tiempo como se muestra en la figura:

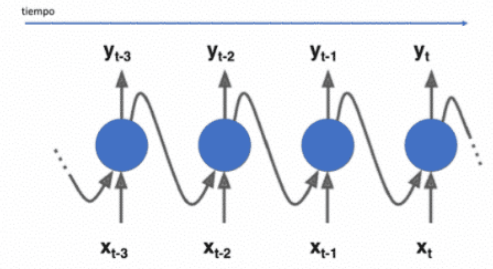

Figura 2. Varias Neuronas en secuencia

Siguiendo esta misma idea, una capa de neuronas recurrentes se puede implementar de tal manera que, en cada instante de tiempo, cada neurona recibe dos entradas, la entrada correspondiente de la capa anterior y su vez la salida del instante anterior de la misma capa.

II-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 ’el’, 3 . ’L’ probablemente también debería ser dado el contexto de ’hel’, y finalmente 4. ’o’ probablemente debería ser dado el contexto de ïnfierno”.

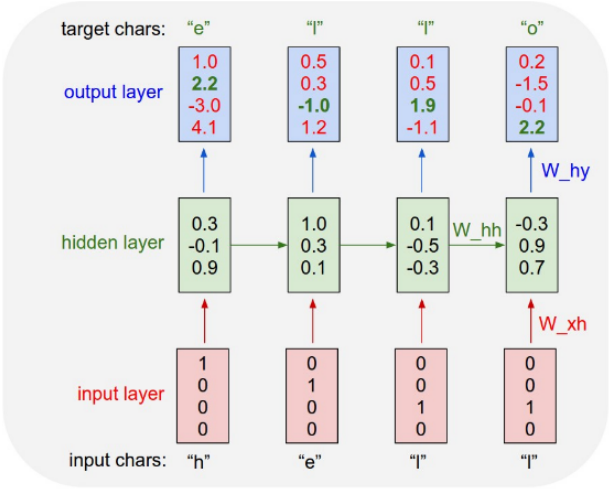

Figura 3. ejemplo de RNN con capas de entrada y salida de 4 dimensiones y una capa oculta de 3 unidades (neuronas)

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 caracteres ïnfierno”. En el otro caso, que es el de salida, esta contiene confedencias en la RNN,la misma que se le asigna para el siguiente caracter.

II-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.

III. Estado Del Arte

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

En este artículo con el mismo nombre Redes Neuronales Recurrentes Para El Análisis De Secuencias [4] 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.

III-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 [5] 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.

III-C. Redes Neuronales Convolucionales y Redes Neuronales Recurrentes

En este artículo de nombre Redes neuronales convolucionales y redes neuronales recurrentes en la transcripción automática [6] 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.

IV. Metodología

Detallaremos las herramientas y metodologías que se usarán para el desarrollo de nuestra red neuronal recurrente LSTM a nivel de carácter:

IV-A. Numpy

Es una libreria de python [7], que permite la creación de cálculos, usada frecuentemente en operaciones con matrices el cual nos será muy útil para el desarrollo de nuestra RNN.

IV-B. PyTorch

PyTorch [9] 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.

IV-C. Métricas

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

IV-D. Procesar la Data

Las redes neuronales solo aceptan números, entonces vamos a tener que hacer que nuestros datos sean números y podemos hacer 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.

IV-E. Haciendo mini-batches para entrenamiento

Para entrenar la data, vamos a crear mini-batches para 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.

IV-F. Entrenamiento

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

IV-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.

Nuestro modelo usa una **softmax**, pero podemos añadir algo de aleatoridad para escoger uno de los caracteres más probables con **Top K** [10], asi puede generar un texto algo aleatorio y aveces absurdo.

IV-H. Metodologia del trabajo

- Cargamos cualquier archivo de extensión .txt que almenos tenga 1000000 caracteres.
- Creamos mini-lotes(mini-batches) para para el entrenamiento. Nuestro lote(batch) debe tener múltiples secuencias de una cantidad de números.
- Definimos nuestra RNN LSTM a nivel caracter con Pytorch.
- Diseñamos nuestra función de entrenamiento el cual va a definir el número de epocas(epochs), learning reate y otros parámetros.
- Una vez que el modelo haya sido entrenado definimos nuestra función para predecir el siguiente carácter, donde usaremos top K para añadir algo de aletoriedad para escoger uno de los caracteres más probables.
- Evaluamos nuestro algoritmo pasando un texto inicial y luego esperando que el nos un texto de longitud especificado.

V. Experimentación y resultados

La experimentación se ejecutó en Google Colab en un entorno de GPU y si el proceso de entrenamiento se hace en CPU el número de épocas tiene que ser pequeño.

V-A. Conjunto de datos

Se uso un archivo .txt de nombre archivo.txt que contiene el resumen de la novela Cien Años de Soledad y el texto dentro de ese archivo tiene una longitud de 691699.

V-B. Tarea: Generación de texto

En este caso para poder generar el texto una vez hecho el diseño del código en python usando mayormente las librerías Torch y Numpy; después de su entrenamiento hacemos el uso de la función **sample()** donde pasamos los siguiente parámetros la red, la cantidad de texto a generar(es decir la longitud del texto a generar), texto inicial y top_k. Obteniedo lo mostrado en la Figura 4:

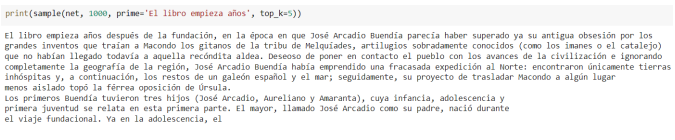

Figura 4. Texto generado después de pasarle como texto inicial **El libro empieza años**

Como podemos observar en la imagen se generó el texto a partir de un texto inicial y el texto generado es de longitud 1000 como específica al momento de usar la función **sample()** pasándole los parámetros mencionados anteriormente.

VI. Discusión De Resultados

VII. Conclusiones

VIII. Trabajos Futuros

Referencias

[1] Inteligencia Artificial: https://www.oracle.com/mx/artificial-intelligence/what-is-ai/
[2] Redes Neuronales: https://www.ibm.com/docs/es/spss-modeler/SaaS?topic=networks-neural-model
[3] Redes Neuronales Recurrentes: https://www.diegocalvo.es/red-neuronal-recurrente/
[4] 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.
[5] Guridi Mateos, Guillermo. Modelos de redes neuronales recurrentes en clasificación de patentes. BS thesis. 2017.
[6] Prieto, Juan Sebastian Gelvez. Redes neuronales convolucionales y redes neuronales recurrentes en la transcripción automática."
[7] Numpy: https://aprendeconalf.es/docencia/python/manual/numpy/
[8] LSMT: https://www.codificandobits.com/blog/redes-lstm/
[9] PyTorch: https://searchenterpriseai.techtarget.com/definition/PyTorch
[10] Top K: https://pytorch.org/docs/stable/torch.htmltorch.topk
[11] https://karpathy.github.io/2015/05/21/rnn-effectiveness/