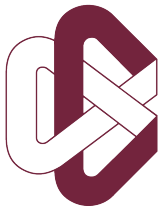

CIMAT

Centro de Investigación en Matemáticas
Unidad Monterrey

Análisis de Texto e Imágenes

Generación y Clasificación de Texto con Deep Learning

Gustavo Hernández Angeles

6 de octubre de 2025

Índice

1 Parte A: Generación de Texto **3**
1.1 Datos 3
1.1.1 Recopilación de datos 3
1.1.2 Limpieza y preprocesamiento del texto 4
1.2 Transformers 5
1.2.1 Evaluación de la Generación de Texto 6
1.3 Modelos RNN, LSTM & GRU 10

Parte A: Generación de Texto

Datos

En esta sección se explica el proceso de recopilación y limpieza de datos para entrenar los modelos de generación de texto utilizando letras de canciones.

Recopilación de datos

Se utilizaron dos scripts de Python para scrapear las letras de las canciones de un artista específico desde la plataforma *Genius*. El primer script (`scrape_songs_alpaca.py`) realiza las siguientes tareas:

- Autenticación en la API de Genius.
- Búsqueda de canciones (URLs) del artista.
- Extracción de las letras de las canciones.
- Limpieza y preprocesamiento de las letras en formato Alpaca para Fine-Tuning.
- Almacenamiento de las letras en un archivo de texto.

El segundo script (`preprocess_scraped_songs.py`) se encarga de procesar la salida del primer script para generar un archivo .txt con el formato adecuado para el entrenamiento de los modelos RNN/LSTM/GRU.

La selección de artistas y el número de canciones a scrapear se encuentran configuradas directamente en el código del primer script. En este caso, elegí a *Kendrick Lamar*, *Kanye West* y *Jay-Z* debido al género compartido *Hip-Hop/Rap* y la riqueza lírica de sus canciones. El número de canciones por artista se estableció en 100, lo que da un total aproximado de 300 canciones.

La ejecución de ambos scripts se realiza desde la terminal con la serie de comandos:

```
python ./codigo/generative/scraping/scrape_songs_alpaca.py
python ./codigo/generative/scraping/preprocess_scraped_songs.py
```

Nota: Es importante asegurarse de tener la clave de API de Genius configurada en un archivo `.env` en el mismo directorio que el script. Ejemplo: `GENIUS_API_TOKEN="tu_token_aqui"`.

Al finalizar la ejecución, obtenemos distintos archivos para ambos formatos de salida particionados en conjuntos de entrenamiento y prueba. Además, también se generan los diccionarios necesarios para el preprocesamiento de texto en los modelos RNN/LSTM/GRU, y estadísticas del conjunto de datos. Los archivos generados son almacenados en el directorio `./data/text_gen/` y son los siguientes:

- `train_lyrics_alpaca.json` y
- `test_lyrics_alpaca.json` (formato Alpaca).
- `train_lyrics.txt` y
- `test_lyrics.txt` (formato para RNN/LSTM/GRU).
- `vocab_char.json` y

- `vocab_word.json` (diccionarios de caracteres y palabras).
- `dataset_info.json` (estadísticas del conjunto de datos).

Limpieza y preprocesamiento del texto

Genius provee las letras en formato HTML dentro de `lyrics containers`, los cuales son etiquetas con el atributo `data-lyrics-container="true"`. El script extrae el texto de estas etiquetas y realiza las siguientes operaciones de limpieza:

- Eliminación de etiquetas HTML.
- Eliminación de líneas en blanco y espacios innecesarios.
- Crea cada registro en formato Alpaca para el fine-tuning.

He decidido no eliminar las anotaciones de las canciones (como [Coro], [Verso 1], etc.) ya que pueden proporcionar contexto adicional al modelo durante el entrenamiento, además de que pueden ser útiles para la generación de texto (haciendo que el modelo también genere anotaciones). Además, se conservaron los metadatos como nombre de la canción y artista, ya que pueden ser útiles para futuras referencias o análisis.

Al realizar Fine-Tuning sobre el LLM, es conveniente proporcionar contexto adicional (y acorde) al modelo. Debido a que en este caso nos decidimos por utilizar un modelo instructivo, el formato Alpaca es adecuado para este propósito. Este formato contiene campos para la instrucción, entrada y salida, lo que permite al modelo aprender a generar texto basado en instrucciones específicas de un solo turno.

```
{
    "instruction": "<INSTRUCCIÓN>",
    "input": "<ENTRADA>",
    "output": "<SALIDA>"
}
```

donde:

- `<INSTRUCCIÓN>` es una cadena que indica la tarea a realizar, en este caso: “You are a hip-hop artist, helping people write lyrics.” o variantes.
- `<ENTRADA>` es una cadena que proporciona contexto adicional, en este caso: “Help me write a song in the style of `<ARTISTA>`.”
- `<SALIDA>` es la letra de la canción.

Por otro lado, al utilizar modelos RNN/LSTM/GRU basta con simplemente tener las letras en texto plano, ya que estos modelos no requieren el mismo nivel de contexto que los LLMs. Sin embargo, es importante asegurarse de que el texto esté limpio y bien formateado para evitar problemas durante el entrenamiento. El script de preprocesamiento convierte los textos de formato Alpaca a texto plano, asegurándose de que cada letra esté separada por los delimitadores de cada canción (En este caso `<|song_start|>` y `<song_end>`).

Transformers

En este trabajo se utilizó el modelo `unsloth/mistral-7B-instruct-v0.3-bnb-4bit` como base para el fine-tuning. Este modelo es una variante del modelo creado por Mistral: Mistral-7B [1], y optimizado por *Unsloth* para tareas de instrucción y cuantizado a 4 bits para reducir el tamaño del modelo.

El framework utilizado para el fine-tuning fue `transformers` de Hugging Face, junto con `peft` para la implementación de Low-Rank Adaptation (LoRA). La elección de LoRA se debe a su eficiencia en términos de memoria y tiempo de entrenamiento, permitiendo adaptar grandes modelos preentrenados con un costo computacional reducido [2].

Tokenizador y representación. El modelo base emplea un tokenizador sub-palabras basado en SentencePiece con un vocabulario de aproximadamente 32K tokens [3]. Este enfoque permite manejar adecuadamente la variabilidad léxica propia de letras de rap (slang, contracciones y variaciones estilísticas) sin inflar excesivamente el vocabulario. No se realizó entrenamiento de un tokenizador nuevo, ya que el objetivo fue un ajuste fino instructivo y no la adaptación desde cero a un dominio léxico extremadamente distinto.

Formato de datos. Cada ejemplo del conjunto de entrenamiento se construyó en formato tipo *Alpaca*: `(instruction, input, output)` y posteriormente convertido a un *chat template* estilo `alpaca` mediante `tokenizer.apply_chat_template`. Para cada canción se genera típicamente un par instrucción–respuesta, donde la instrucción solicita la generación de letras en el estilo de un artista específico y la salida contiene la letra correspondiente. Tras el mapeo, el campo final consumido por el entrenador es un único texto ya formateado.

Tamaño del conjunto. El corpus curado contiene 239 ejemplos de entrenamiento y 60 de prueba (correspondientes a canciones únicas). Con un tamaño de lote efectivo de 16 secuencias (lote por dispositivo = 2, acumulación de gradientes = 8), el número de pasos por época es:

$$\text{steps/epoch} = \lceil 239/16 \rceil = 15.$$

Se entrenaron 3 épocas, resultando en 45 pasos totales (se almacenaron checkpoints intermedios consistentes con este valor). Esto refleja un escenario de ajuste fino ligero, más orientado a estilo que a aprendizaje exhaustivo.

Longitud de secuencia. Se fijó `max_seq_length = 2048`. Dado que las letras individuales rara vez exceden este límite, se deshabilitó el *packing* (`packing=False`) para evitar mezclar canciones dentro de la misma secuencia y preservar coherencia estructural (secciones como [Verse], [Chorus], etc.).

Configuración LoRA. Se aplicó LoRA con `r = 32`, `lora_alpha = 64`, `lora_dropout = 0.0` sobre los módulos de proyección de atención y MLP: `q_proj, k_proj, v_proj, o_proj, gate_proj, up_proj, down_proj`. Suponiendo dimensiones del modelo base (d_model = 4096, tamaño intermedio = 14336, 32 capas), el número aproximado de parámetros entrenables añadidos por LoRA es:

$$\text{por capa} = 4 \times r(4096+4096) + 3 \times r(4096+14336) = 4(32 \cdot 8192) + 3(32 \cdot 18432) \approx 2{,}818{,}048,$$

$$\text{total} \approx 32 \times 2{,}818{,}048 \approx 90.2\text{M},$$

lo que representa 1.2 % de los =7.3B parámetros totales, pero concentra toda la capacidad de adaptación. Estos pesos LoRA se mantienen en precisión media (FP16/BF16) mientras los pesos base permanecen cuantizados a 4 bits [4].

Hiperparámetros de entrenamiento. Se utilizaron: tasa de aprendizaje `2e-4` (scheduler lineal con calentamiento de 5 pasos), `epochs = 3`, `per_device_train_batch_size = 2`, `gradient_accumulation_steps = 8`, `weight_decay = 0.01`, optimizador `AdamW 8-bit` [5], y semilla `3407`. El tamaño de lote efectivo resultante es 16 ejemplos. No se aplicó regularización adicional (`lora_dropout = 0`).

Gestión de memoria y hardware. El uso combinado de cuantización 4-bit + LoRA reduce el pico de VRAM necesario a un rango que típicamente cabe en una sola GPU de 12–16 GB manteniendo la ventana de contexto de 2048. El script registra memoria reservada y máxima por dispositivo antes y después del entrenamiento.

Hardware utilizado. El entrenamiento se realizó en un nodo con 2 GPUs NVIDIA RTX TITAN (24 GB VRAM cada una), 128 GB RAM y 24 núcleos de CPU Intel Xeon Silver 4214 (2.2 GHz). Equipo proporcionado por el Laboratorio de Supercómputo del Bajío (Lab-SB) del CIMAT.

Evaluación de la Generación de Texto

Perplejidad. La *perplejidad* (PPL) es una métrica estándar para evaluar modelos de lenguaje, que mide qué tan bien un modelo predice una muestra. Matemáticamente, la perplejidad se define como la exponencial de la entropía cruzada promedio por token:

$$\mathrm{PPL} = \exp\left(-\frac{1}{N}\sum_{i=1}^{N}\log P(w_i|w_{<i})\right),$$

donde N es el número total de tokens en el conjunto de evaluación, y $P(w_i|w_{<i})$ es la probabilidad condicional del token w_i dado el contexto previo $w_{<i}$. Una perplejidad más baja indica un mejor rendimiento del modelo, ya que sugiere que el modelo está menos "sorprendido"por los datos de evaluación.

Para evaluar el impacto del fine-tuning, se comparó la perplejidad del modelo ajustado con la del modelo base `mistral-7B-instruct-v0.3-bnb-4bit` sin ajuste fino. Esto permite cuantificar la mejora en la capacidad predictiva del modelo tras el entrenamiento con las letras de canciones. El código para calcular la perplejidad es proporcionado por la misma biblioteca `Unsloth`, se encuentra en `codigo/generative/mistral/utils/perplexity_unsloth.py`. La lógica fue adaptada en el script `calculate_perplexity.py` para evaluar ambos modelos (base y fine-tuned) sobre el conjunto de prueba.

El resultado de la evaluación de perplejidad en el conjunto de prueba es el siguiente:

- Perplejidad del modelo base (sin fine-tuning): 9.35.
- Perplejidad del modelo fine-tuned: 7.20.
- Mejora relativa: $(9.35 - 7.20)/9.35 \times 100\,\% \approx 22.93\,\%$.
- Interpretación: La reducción significativa en la perplejidad indica que el modelo fine-tuned ha mejorado su capacidad para predecir las letras de canciones, adaptándose mejor al estilo y vocabulario específico del dominio. Esto fue logrado a pesar del tamaño relativamente pequeño del conjunto de datos de entrenamiento, demostrando la eficacia del enfoque de fine-tuning con LoRA en un modelo cuantizado.

Muestras generadas.

Se generaron varias muestras de texto utilizando el modelo fine-tuned para evaluar su capacidad de generación. Todas las muestras se generaron con los siguientes parámetros en la inferencia, por mostrar un mejor balance entre coherencia y diversidad:

- `max_new_tokens = 256`. Representa la longitud máxima de la secuencia generada.
- `temperature = 0.8` (0.7 para Kendrick Lamar). Controla la aleatoriedad de la generación; valores más bajos hacen que el modelo sea más conservador.
- `top_p = 0.9`. Utiliza muestreo de núcleo para limitar la selección de tokens a los más probables.
- `repetition_penalty = 1.3`. Penaliza la repetición de tokens para fomentar la diversidad en la generación.

A continuación se presentan algunos ejemplos representativos:

Ejemplo 1: Letra generada en el estilo de Jay-Z

En este ejemplo, se le pidió al modelo que generara una letra en el estilo de Jay-Z mediante el prompt:

```
You're a hip-hop artist, create song lyrics.
Write hip-hop lyrics in the style of Jay-Z.
```

Y se obtuvo la siguiente salida:

```
[Intro]
Uh huh uh (yeah)
Yeah yeah you know my name is Hov', right? Uhh
I got this little thing called ``The Blueprint''
And I just want to run through some shit for y'all tonight
Just have fun with it
(Hands up!)

[Verse 1]
Yo, yo, yo, yo, whattup! It's your boy Shawn
Back on that bullshit 'cause niggas ain't learn
Last time out I burned down the barn
Now I'm back and I might burn down another one
What do you wanna bet? You can put all ya chips in
If he don't win then we all gonna be broke again
But if he does win then we gon' get richer than rich men
So place your bets now before they stop taking them
See me as an investment like Berkley or Microsoft
Put ten dollars into me baby watch how far it goes
Watch how fast it grows
```

Podemos analizar la letra generada entre pilares del estilo de cada artista:

- **Persona y Autenticidad.** El intro es muy representativo del estilo de Jay-Z, utilizando *ad-libs* (Uh, uh, yeah) y directamente referenciando su nickname (Hov'). También hace

referencia a su álbum “The Blueprint”, lo que refuerza su identidad como artista. El uso de “Your boy Shawn” es también auténtico.

- **Temas y Contenido.** La letra realiza una metáfora sobre invertir en sí mismo como un artista, lo cual es un tema recurrente en la música de Jay-Z.
- **Rimas y Fluidez.** La letra presenta un esquema de rima consistente (AABB), aunque la estructura es predecible y carece de complejidad y expertiz de Jay-Z.

Ejemplo 2: Letra generada en el estilo de Kanye West

Nuevamente, se le pidió al modelo la generación de una letra, ahora en el estilo de Kanye West. Esto fue mediante el prompt:

```
You’re a hip-hop artist, create song lyrics.
Write hip-hop lyrics in the style of Kanye West.
```

Y se obtuvo la siguiente salida:

```
[Intro]
I know He won’t forsake me
He won’t leave me
No matter what I do
And even if I go through hell tonight
He will still be God when it’s all over with
(God is) (God is)

[Verse 1]
They say build your own
I said fuck that, we gon’ buy our way out
We ain’t finna wait ’til they give us something to celebrate
So let’s get this bread
Fuck up school and just drop out
But you can’t drop out without dropping some knowledge first
Let me teach y’all how to make yourself rich
How to stay fly while staying righteous
First thing you need is faith as big as Mount Rushmore
Then take it from here

[Chorus]
Jesus saved my soul
Now I drive Rollies and Bentleys
Bought myself a couple chains
Spendin’ money like I don’t owe Him nothin’
Everybody want heaven but nobody wanna die
Nobody wanna cry no more
That means everybody wanna live forever
Not ready to meet Your maker
```

- **Persona y Autenticidad.** La letra captura la dualidad central de la persona de Kanye West: la del predicador y el materialista. El intro con temática góspel y las referencias

a Dios son muy representativas de su era en los finales de los 10's con el album *Jesus is King*. Esto se contrasta inmediatamente con un tono provocador y egocéntrico (Let me teach y'all", fuck that"), lo cual es sumamente auténtico a su estilo.

- **Temas y Contenido.** El modelo representa exitosamente la mezcla de temas de Kanye. Se abordan la religión y la fe ("Jesus saved my soul", "faith as big as Mount Rushmore") en conflicto directo con el materialismo extremo y el lujo ("drive Rollies and Bentleys"). Incluye también aforismos y comentarios sociales simplificados, como la línea "Everybody want heaven but nobody wanna die", un recurso muy común en sus letras.
- **Rimas y Fluidez.** Kanye a menudo prioriza el mensaje y la emoción sobre la complejidad de la rima. La letra generada refleja esto: el esquema de rima es inconsistente y conversacional. Se apoya más en la cadencia y en la fuerza de las declaraciones que en patrones de rima complejos, lo que resulta en un flujo que se asemeja a un sermón o una declaración apasionada, muy fiel a su estilo de entrega.

Ejemplo 3: Letra generada en el estilo de Kendrick Lamar

Esta vez se le pidió al modelo generar una letra para una canción de Hip-Hop al estilo de Kendrick Lamar, mediante el prompt:

```
You're a hip-hop artist, create song lyrics.
Write hip-hop lyrics in the style of Kendrick Lamar.
```

Y se obtuvo la siguiente salida:

```
[Intro]
I don't know what they want from me! I gave them my life
They wanna see some type of drama? Here it is
(Bitch)

[Verse 1]
What up homie? What you writing about today? (Today?)
Writing 'bout these niggas that be hating on me (Huh?)
And all this fake love that surrounds me (Uh)
All this bullshit that goes around me (Yeah)
It gets tiring sometimes when you trying to stay positive
But negative energy always surrounding you
So how can we live with no stress and anxiety?
When everybody got something bad to say bout you
Everybody wants something but ain't willing to give nothin' back
Nobody really cares if you win or lose at shit
If your name isn't mentioned then nobody gon' mention you
Even though you did more than most people ever done
That'll make you feel like you worthless
Like you never won
Never been number one
Always second place even when you first
How do you deal with that kind of pressure? How do you cope?
Knowin' every move you make will affect your whole family tree
```

```
Your friends, your enemies, your lovers, haters, neighbors
The world, society, everyone expect perfection
From an imperfect person
```

- **Persona y Autenticidad.** Captura a la perfección la persona introspectiva, ansiosa y abrumada que define gran parte de la obra reciente de Kendrick Lamar (DAMN y MM&BS). La letra se lee como un monólogo interno o una sesión de terapia, explorando la presión de la fama de una manera vulnerable y conflictiva, lo cual es una característica central de su estilo.
- **Temas y Contenido.** Los temas abordados son extremadamente fieles al universo lírico de Kendrick. Se centra en la salud mental ("stress and anxiety"), el peso de las expectativas ("everyone expect perfection"), la falsedad en la industria ("fake love that surrounds me"), y la responsabilidad generacional ("every move you make will affect your whole family tree"). Este enfoque en el conflicto interno y la carga del liderazgo es mucho más representativo que la simple bravuconería.
- **Rimas y Fluidez.** El modelo imita exitosamente la fluidez conversacional y casi de "spoken word"de Kendrick. En lugar de adherirse a un esquema de rima estricto, prioriza el flujo de conciencia y la carga emocional del mensaje. Las líneas son largas y discursivas, construyendo una tensión que culmina en la lista final ("Your friends, your enemies..."). Esta estructura, que favorece el ritmo narrativo sobre la rima perfecta, es una técnica clave en el repertorio de Kendrick.

En los tres ejemplos, el modelo fine-tuned demuestra una capacidad notable para capturar los estilos únicos de cada artista, tanto en términos de contenido temático como la estructura lírica. Aunque no alcanza la complejidad y profundidad de los artistas originales, especialmente en términos de rima y metáforas, el modelo genera letras coherentes y estilísticamente apropiadas para los artistas. Además, el modelo respetó en su mayoría la estructura de las letras basandose en los datos de entrenamiento (por ejemplo, siempre inicio con el [Intro], [Verse 1], etc.). Esto sugiere que el enfoque de fine-tuning con un conjunto de datos relativamente pequeño pero bien curado puede ser efectivo para adaptar un modelo de lenguaje grande a tareas creativas específicas como la generación de letras de canciones.

Modelos RNN, LSTM & GRU

Referencias

[1] Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mistral 7b, 2023.

[2] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3, 2022.

[3] Taku Kudo and John Richardson. Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. *arXiv preprint arXiv:1808.06226*, 2018.

[4] Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. Qlora: Efficient finetuning of quantized llms, 2023.

[5] Tim Dettmers, Mike Lewis, Sam Shleifer, and Luke Zettlemoyer. 8-bit optimizers via blockwise quantization. *9th International Conference on Learning Representations, ICLR*, 2022.