Universidad Autónoma Metropolitana

Posgrado en Ciencia y Tecnologías de la Información

Generación Automatizada de Textos: Poesía

Anthony Pérez Rangel
Matrícula: 2202800399

Asesores

Dr. Sergio Gerardo de los Cobos Silva (UAM-I)

Dr. Roman Anselmo Mora Gutiérrez (UAM-A)

Resumen

La Generación de Lenguaje Natural (GLN o NLG por sus siglas en inglés) se puede entender como un subcampo del Procesamiento del Lenguaje Natural (PLN) donde la diferencia es el enfoque sobre el lenguaje, la GLN consiste en crear de forma automatizada texto con el fin de comunicarse. Este texto puede tener diversas aplicaciones cómo, la generación de informes, por ejemplo, para la generación de noticias o informes del clima, la traducción entre idiomas, los sistemas de diálogo, como los Chatbots y asistentes personales, la generación de historias, el mejor ejemplo es el videojuego Dungeon AI, que genera una historia de acuerdo a los parámetros dados por el usuario, como género literario, ambiente, protagonista, etcétera, la generación de poesía, la generación de texto con humor, etc.
En este trabajo se presenta el caso particular de la generación de poesía en español, es una tarea que combina diversas ramas del conocimiento, como son la Inteligencia Artificial, la Lingüística, la Literatura, la Semántica, la Gramática, y la Creatividad Computacional, entre otras. Se decidió por el idioma español debido a que existen menos trabajos que en lenguajes más populares como el inglés o el chino. Se usaron modelos de redes neuronales recurrentes, que son generalmente empleados para tareas del PLN. También se utilizaron los modelos de redes neuronales de mayor potencia en la actualidad conocidos como Transformers, para tratar el problema y mostrar el potencial del uso de estos modelos masivos. Entre las aportaciones de este trabajo está la creación de una base de datos desde cero de más de 10 mil poemas en español, exclusivamente desarrollada para el problema. También el desarrollo de modelos de Redes neuronales, usando Long Short Term Memory (LSTM) y Gated Recurrent Unit (GRU), uso de Embeddings como Word2Vec para la codificación del lenguaje y por último, el entrenamiento y uso de un modelo Transformer, en este caso se usó GPT-2, siendo la primera vez que se usa este tipo de modelos para la tarea de generación de poesía en español.

Índice general

1. Introducción **5**
- 1.1. Aspectos generales . . . 5
- 1.2. Aplicaciones . . . 6
- 1.3. Planteamiento del problema . . . 6
- 1.4. Justificación . . . 7
- 1.5. Objetivos . . . 7
  - 1.5.1. Objetivo general . . . 7
  - 1.5.2. Objetivos particulares . . . 7
- 1.6. Metodología . . . 7
- 1.7. Redes neuronales . . . 8
  - 1.7.1. Redes Neuronales Recurrentes . . . 8
  - 1.7.2. Transformers . . . 8

2. Estado del arte **10**

3. Experimentos **12**
- 3.1. Resultados primera fase: Redes Neuronales Recurrentes . . . 12
  - 3.1.1. Base de datos . . . 13
  - 3.1.2. Proceso de entrenamiento de los 2 modelos . . . 13
  - 3.1.3. Resultados generación de poesía a nivel caracter . . . 14
  - 3.1.4. Resultados generación de poesía a nivel palabra . . . 18
- 3.2. Resultados Segunda fase: Transformers . . . 22
  - 3.2.1. Base de datos y Modelo . . . 22
  - 3.2.2. Proceso de entrenamiento de los modelos . . . 23
  - 3.2.3. Experimentos y resultados . . . 24
- 3.3. Encuesta . . . 31
  - 3.3.1. Descripción . . . 31
  - 3.3.2. Universo muestral . . . 32
  - 3.3.3. Medio Utilizado . . . 32
  - 3.3.4. Métrica . . . 32
  - 3.3.5. Resultados de las encuestas . . . 32

4. Conclusiones **34**

A. Inteligencia Artificial **39**
- A.1. ¿Qué es la Inteligencia Artificial? . . . 39
  - A.1.1. Antecedentes . . . 39
  - A.1.2. Aplicaciones de la Inteligencia Artificial . . . 43
  - A.1.3. Machine learning . . . 43
  - A.1.4. Tipos de algoritmos . . . 44
  - A.1.5. ¿Qué es Deep learning? . . . 45
  - A.1.6. Diferencias Machine Learning y Deep Learning . . . 45

B. Redes neuronales **46**
B.1. ¿Qué es una red neuronal? . . . 46
B.2. Elementos de una red . . . 46
B.2.1. Perceptrón . . . 46
B.2.2. Capas . . . 47
B.2.3. Pesos . . . 48
B.2.4. Bias o sesgo . . . 48
B.2.5. Función de Activación . . . 48
B.2.6. Función de coste . . . 49
B.2.7. Backpropagation . . . 49
B.3. Redes Neuronales Convolucionales . . . 52
B.4. Redes Neuronales Recurrentes . . . 53
B.5. Redes Long Short-Term Memory . . . 53
B.6. Redes Gated Recurrent Unit . . . 56

C. Transformers **58**
C.1. Transformers . . . 58
C.1.1. Arquitectura . . . 59
C.1.2. Input Embedding y Positional Encoding . . . 59
C.1.3. Mecanismo Self-Attention . . . 60
C.1.4. Encoder . . . 62
C.1.5. Decoder . . . 63
C.2. ¿Qué es GPT-2? . . . 64
C.2.1. Características . . . 65
C.2.2. Arquitectura . . . 66
C.2.3. Masked Self-Attention . . . 67
C.2.4. Byte pair Encoding (BPE) . . . 67

D. Transfer learning **71**
D.1. Transfer Learning o Transferencia de Conocimiento . . . 71
D.1.1. Fine tuning . . . 71
D.1.2. Ajustes en Fine Tuning . . . 72

E. Poesía **74**
E.1. Elementos de la Poesía . . . 74
E.1.1. Prosa . . . 74
E.1.2. Verso . . . 74
E.1.3. Métrica . . . 74
E.1.4. Licencia Poética . . . 75
E.1.5. Rima . . . 75
E.1.6. Figuras Literarias . . . 75
E.1.7. Géneros poéticos principales . . . 76

F. Encuestas **77**
F.1. Cuestionario . . . 77
*

Índice de figuras

3.1. Forma de un poema en la base de datos 13

B.1. Diagrama de un Perceptrón simple 47
B.2. Diagrama de una Red Neuronal 48
B.3. Visualización del descenso del gradiente 50
B.4. Ejemplo de Red neuronal Convolucional 52
B.5. Ejemplo de Red neuronal Recurrente 53
B.6. Modelo de compuertas en una Red LSTM 54
B.7. Modelo de compuertas en una Red GRU 56

C.1. Conexión amplificada entre en Encoder y un Decoder 59
C.2. Vista matricial del Embedding 60
C.3. Vista del Encoding Posicional 61
C.4. Arquitectura de un Módulo de Atención 61
C.5. Producto punto con cada palabra 62
C.6. Contenido interno del primer Encoder en el bloque 63
C.7. Contenido interno de un Decoder en el bloque 64
C.8. Matriz que representa la relación entre elementos en el mecanismo Self-Attention 65
C.9. Apariencia de un decoder en GPT-2 66
C.10. Matriz enmascarada en el módulo Masked Self-Attention 67

Índice de cuadros

3.1. Resultado experimento 1 de generación por caracter 15
3.2. Resultado experimento 2 de generación por caracter 16
3.3. Resultado experimento 3 de generación por caracter 16
3.4. Resultado experimento 4 de generación por caracter 17
3.5. Resultado experimento 5 de generación por caracter 18
3.6. Resultado experimento 1 de generación por palabras 19
3.7. Resultado experimento 2 de generación por palabras 19
3.8. Resultado experimento 3 de generación por palabras 20
3.9. Resultado experimento 4 de generación por palabras 21
3.10. Resultado experimento 5 de generación por palabras 21
3.11. Resultado experimento 6 de generación por palabras 22
3.12. Poema 1 Experimento 1 de generación con Transformers 25
3.13. Poema 2 Experimento 1 de generación con Transformers 25
3.14. Poema 3 Experimento 1 de generación con Transformers 26
3.15. Poema 1 Experimento 2 de generación con Transformers 26
3.16. Poema 2 Experimento 2 de generación con Transformers 27
3.17. Poema 3 Experimento 2 de generación con Transformers 27
3.18. Poema 1 Experimento 3 de generación con Transformers 28
3.19. Poema 2 Experimento 3 de generación con Transformers 28
3.20. Poema 3 Experimento 3 de generación con Transformers 28
3.21. Poema 1 Experimento 4 de generación con Transformers 29
3.22. Poema 2 Experimento 4 de generación con Transformers 29
3.23. Poema 3 Experimento 4 de generación con Transformers 30
3.24. Poema 1 Experimento 5 de generación con Transformers 30
3.25. Poema 2 Experimento 5 de generación con Transformers 31
3.26. Poema 3 Experimento 5 de generación con Transformers 31

C.1. Tabla 1 Ejemplo de BPE 68
C.2. Tabla 2 Ejemplo de BPE 68
C.3. Tabla 3 Ejemplo de BPE 68
C.4. Tabla 4 Ejemplo de BPE 69
C.5. Tabla 5 Ejemplo de BPE 69
C.6. Tabla 6 Ejemplo de BPE 70

Capítulo 1

Introducción

1.1. Aspectos generales

¿Qué es la poesía?

La poesía es un género literario que busca expresar los sentimientos y emociones a través del lenguaje y una de las formas de arte más añejas que se conoce. La poesía se ha extendido a prácticamente todas las lenguas como, por ejemplo, inglés, chino, ruso, portugués, español, etc. Cada uno con sus estilos y características que los hacen únicos.
Una de las características principales es la forma en que se escribe, ya sea que esté escrita en verso, la forma más conocida donde el texto se divide en renglones cortos de pocas sílabas o en prosa, parecida al texto que se usa de forma cotidiana, pero usando diversos recursos literarios como metáforas, hipérboles, figuras retóricas, etc.
La poesía no está limitada a los poemas (aunque es la forma de manifestación más común en la poesía) y se puede encontrar en canciones, cuentos, relatos o cualquier otra obra literaria que expresa la belleza al ser leída o escuchada, siendo cada obra un estilo distinto e independiente para cada escritor.
También se ha utilizado para relatar la historia de muchas culturas y sus tradiciones, siendo la cultura china uno de los mejores ejemplos con los poemas de Da Fu que narran hechos históricos de su tiempo.
No existe un consenso de cómo crear poesía, debido a que es un género motivado por la inspiración que puede ser abordado desde distintas perspectivas y con distintas concepciones de belleza, por lo que los estilos son únicos y complicados de imitar.
Los textos poéticos son distintos a otro tipo de escritos, como los textos informativos, por ejemplo, las noticias, las crónicas, las biografías, los diccionarios o los manuales, donde el principal objetivo es brindar información directa y concisa. En cambio, el texto poético tiene como propósito, ser el medio principal donde el autor busca plasmar su inspiración en forma de palabras, abordando sus sentimientos y emociones, utilizando un sin fin de temas para moldear todas sus ideas.

¿Qué es el Procesamiento del Lenguaje Natural y la Generación del Lenguaje Natural?

El Procesamiento del Lenguaje Natural (PLN), es el campo de la inteligencia artificial encargado de estudiar la forma en que las máquinas se comunican utilizando el lenguaje natural, también conocido como lenguaje humano. Estudia el procesamiento, la comprensión y la generación del lenguaje, abarcando también campos como la lógica y la lingüística computacional.
En cuanto a la Generación del Lenguaje Natural (GLN), su función consiste en que la máquina genere de forma automatizada textos con el fin de comunicarse, a partir de datos de entrada como pueden ser palabras clave, frases, sonidos, imágenes, etc. La máquina

debe de ser capaz de generar texto de cualquier extensión, que tenga sentido y sea acorde con la entrada que el usuario le proporciona.
Entre las aplicaciones que se pueden encontrar a esta tarea van desde la generación de resúmenes, los sistemas de diálogo, la generación automatizada de historias, la generación de poesía, la generación de paráfrasis, la generación de textos con humor y la generación de noticias falsas, entre otras aplicaciones derivadas.

1.2. Aplicaciones

Las aplicaciones de la generación automática de poesía son actividades derivadas de la generación del lenguaje natural pero están enfocadas en la generación de poesía.

- Creación de libros: Se han creado libros de forma automática usando redes neuronales u otros métodos. En alcance va desde libros de historia, hasta libros de recetas pasando por libros de poesía.
- Sistemas asistentes de creación de poesía. Estos asistentes ayudan a la persona a completar obras, van desde completar renglones, hasta poemas completos dependiendo de la complejidad del modelo y con cuanta información haya sido entrenado.

Cabe resaltar que este tipo de tareas han estado en auge en el campo de la Creatividad Computacional, donde el objetivo es generar programas que puedan producir contenido considerado creativo o servir de herramientas para apoyar la creatividad humana. Siendo la poesía una de las manifestaciones creativas por excelencia del ser humano y es una excelente forma para contemplar el alcance actual de las máquinas para la generación de contenido.

1.3. Planteamiento del problema

La generación de poesía es una tarea que ha sido explorada en idiomas muy populares como el inglés y el chino, incluso en otros con menos hablantes como es el coreano, pero en español existen pocos antecedentes de estos estudios. El primer antecedente de la generación de poesía en español fue de Pablo Gervaz [13] con sus sistema WASP (Wishful Automatic Spanish Poet) en el año 2000, un sistema que usa la técnica de moldes, se refiere a tener una base que se va a modificar, en este caso, sería un verso como base e ir sumando, quitando o intercambiando palabras, de esta forma se generan poemas con la misma métrica. Possteriormente aparece ASPERA, una versión refinada que genera resultados más detallados, ambos modelos producen versos basados en la estructura de poemas previos. También se han desarrollado sistemas como Tra-La-Lyrics [14] un programa capaz de generar letras de acuerdo a la melodía de entrada, considerando diversas métricas. PoeTryMe [15], que genera poesía a través de mapas conceptuales. E incluso metodologías para generar poesía *Poet's Little Helpe* [16] centrándose en su forma y contenido. Todos estos centrados en generación en español.
En cambio, en otros idiomas se ha visto el desarrollo de generadores de poesía utilizando diversos modelos de redes neuronales recurrentes, debido a que estos modelos de aprendizaje maquinal tienen la capacidad de aprender patrones y características de las secuencias de datos que reciben, en este caso poemas y con esto permitiendo a los investigadores centrarse en distintos aspectos de la poesía, desde su forma, su estilo o su contenido.
La tendencia actual ha guiado las investigaciones a la búsqueda y el uso de herramientas aún más poderosas que las redes recurrentes, con el propósito en mente de desarrollar modelos que logren generar texto más cercano al de autores humanos, debido principalmente al impedimento que tienen la redes recurrentes para generar textos de gran extensión, con pocos datos de entrada, como puede ser una palabra o un renglón corto y que el texto de

salida mantenga la coherencia.
Una solución popular en la actualidad es el uso de modelos Transformers que ya se ha aplicado en varios idiomas, principalmente el chino, pero no se ha usado para el español.

1.4. Justificación

Como se puede ver, la generación de poesía en español es un área que no se ha explorado tanto como en otros idiomas, a pesar del interés de los investigadores para desarrollarla y hacerla crecer, dando oportunidad a investigaciones de modelos y herramientas que ayuden a expandir el área de la GLN para el idioma español, tomando en cuenta las múltiples aplicaciones que se pueden desarrollar.
Con esto en mente, se propuso el desarrollo de dos modelos para la generación de poesía en español. El primero, utilizando redes neuronales recurrentes, debido a que son la alternativa más popular para resolver problemas de PLN, además de tener una base sólida de información sobre la forma en que se implementan y sus aplicaciones al PLN, para apoyarse en el desarrollo de los modelos para la GLN.
El segundo modelo, está centrado en el uso de las redes neuronales Transformers, que son modelos con una arquitectura completamente distinta a las redes recurrentes, mencionando también que no existen publicaciones de uso de Transformers para la generación de poesía en español, siendo una gran oportunidad de experimentar de primera mano los resultados que se pueden obtener, con las arquitecturas más modernas disponibles en el campo del PLN.

1.5. Objetivos

1.5.1. Objetivo general

Comparar la generación de poesía en español mediante el uso de redes neuronales recurrentes y modelos Transformers.

1.5.2. Objetivos particulares

- Desarrollar un modelo basado en redes neuronales recurrentes para la generación de poesía en español.
- Desarrollar un modelo basado en Transformers para la generación de poesía en español.

1.6. Metodología

La metodología a seguir es la siguiente:

- Revisar los conceptos más importantes sobre redes neuronales básicas.
- Revisar los conceptos más importantes sobre redes neuronales recurrentes, en particular las de tipo Long Short Term Memory (LSTM) y Gated Recurrent Unit (GRU).
- Revisar los conceptos más importantes sobre los Transformers, en particular del modelo GPT-2.
- Revisar los conceptos principales sobre poesía, características y elementos.
- Revisar el estado del arte para seleccionar al menos dos enfoques distintos que puedan aplicarse al problema de interés.

- Desarrollar y entrenar dos algoritmos basados en los modelos seleccionados para la generación de poesía en español.
- Revisar las salidas de las instancias.
- Reportar los resultados en la ICR.

1.7. Redes neuronales

Las redes neuronales son modelos de Machine Learning (ML por sus siglas en inglés) muy extendidos y con múltiples aplicaciones en todos los campos del conocimiento, sus aplicaciones van desde el reconocimiento de rostros, la clasificación de imágenes, la traducción entre idiomas, la generación de noticias, la predicción del clima, la generación de texto con humor, la generación de música, etc.
Una red neuronal pretende imitar el funcionamiento del cerebro humano para el aprendizaje, donde la principal unidad de la que están formadas estas redes es el perceptrón (ver apéndice B), éste se puede entender como la versión artificial de una neurona biológica. Los perceptrones se conectan entre sí para formar redes complejas permitiendo la resolución de múltiples tareas. Normalmente una red neuronal se divide en tres capas, capa de entrada, donde se reciben los datos del problema, la capa oculta, que puede estar compuesta por múltiples neuronas conectadas entre sí y la capa de salida, que se encarga de generar las predicciones para el problema.
Pero existen arquitecturas más complejas enfocadas en procesar datos específicos como las redes recurrentes que se especializan en procesar secuencias de datos, las redes convolucionales que procesan imágenes, etc.
Entre ellas las redes más usadas para la generación de texto son las redes neuronales recurrentes que fueron las primeras enfocadas en la tarea y en la actualidad los Transformers con la capacidad de procesar secuencias pero de una forma distinta.

1.7.1. Redes Neuronales Recurrentes

Para el caso de las redes recurrentes, su principal propósito es trabajar con secuencias de datos como puede ser cualquier texto. Lo hacen simulando una memoria que permite guardar la relación entre los elementos de la secuencia, con el fin de predecir el siguiente elemento (ver el apéndice B).
El principal problema de estos modelos es que para secuencias largas, no es capaz de formar relaciones entre palabras, por un efecto similar a la pérdida de memoria. Durante el entrenamiento ésto genera que la red no logre aprender debidamente dejando estancado el modelo.
Para solucionar el problema surgieron variantes como las redes Long Short Term Memory (LSTM) y Gated Recurrent Unit (GRU) que son capaces de conservar la información que se considere importante para la secuencia y borrando la información que no será necesaria, así evitando la pérdida de información clave para cada etapa durante el entrenamiento.

1.7.2. Transformers

Los Transformers (ver apéndice C), modelos relativamente nuevos, también trabajan con secuencias de datos, pero a diferencia de las redes recurrentes que lo hacen con un elemento a la vez,volviéndose muy lentas y costosas en recursos tecnológicos, estos nuevos modelos pueden trabajar en paralelo con la secuencia completa. Además de ser más veloces al trabajar con toda la secuencia en paralelo permitiendo aprovechar nuevas arquitecturas tecnológicas como las de las Unidades de Procesamiento de Gráficos (GPU por sus siglas

en inglés).
También logran obtener niveles de relación entre las palabras que no es posible alcanzar con las redes recurrentes, debido al uso de técnicas especiales conocidas como *Mecanismos de auto-atención*,
Existe una gran variedad de Transformers que se especializan en diversas tareas, desde los enfocados en la generación de texto como GPT (Generative Pre-trained Transformer) o los enfocados en traducción y clasificación como BERT (Bidirectional Encoder Representations from Transformers), habiendo un modelo para cada necesidad y siendo arquitecturas versátiles, con el único inconveniente de requerir un alto consumo de recursos tecnológicos y una gran cantidad de datos necesarios para el entrenamiento.

Capítulo 2

Estado del arte

En la actualidad las redes neuronales son la herramienta más populares para la generación de texto, gracias a su capacidad de aprender características del lenguaje. Se han desarrollado diversas arquitecturas para tratar el problema de la generación de texto, donde se combinan las redes neuronales recurrentes con otros modelos como autoencoders o redes convolucionales,

En su artículo Astigarraga [16] muestra una metodología para desarrollar un generador de poesía centrado en la forma y el contenido del poema. Separando la generación en tres niveles, el análisis léxico (buscar la rimas similares a la entrada), el análisis semántico (versos similares con rima similar) y la generación de texto(construcción estrofa a estrofa).

El artículo de Wang y Tiang [17] usa un modelo de redes LSTM para desarrollar un modelo que imite el estilo de un poeta en específico. También Eun-Soon, Soohwan y Su-Yeon [21] realizan un generador de poesía pero a diferencia del anterior que sólo puede imitar un estilo, éste es capaz de imitar cuatro estilos distintos, usa redes LSTM para el manejo de las secuencias pero la base es una arquitectura autoencoder que implementa mecanismos de atención (ver apéndice C), para mejorar el aprendizaje de la relación entre palabras, otra diferencia es que la generación es a nivel palabra evitando la generación de palabras desconocidas.

Liu y Fu [18], crearon un modelo capaz de generar poesía tomando como entrada una imagen, esto fue logrado combinando redes GRU, redes Convolucionales y Redes Generativas Adversarias (GAN por sus siglas en inglés), la primera para el procesamiento y generación de secuencias, la segunda para el procesamiento de la imagen y la extracción de palabras clave y la tercera para la selección de los mejores poemas entre los generados. Los poemas son de verso libre,parecido al texto en prosa.

Y. Liu, D. Liu y Lv. [22] crearon un modelo (disponible en Wechat[1]) mucho más complejo, es capaz de generar poesía teniendo como entrada imágenes, fragmentos de un poema y conceptos artísticos, está enfocado en ser un asistente para que las personas puedan generar sus propios poemas e incluso texto acróstico (texto con un mensaje escondido), de parte de la arquitectura solo se menciona el uso de una red neuronal de auto atención, en este caso los modelos que usan éstos mecanismos son los Transformers.

Santillan y A. Azcarraga [25] combinan un Transformers para la generación de múltiples poemas simultáneamente y Doc2Vec para evaluar la relación entre la secuencia de entrada y los poema de salida buscando el mejor poema, se observó que los Transformers pueden aprender estilos específicos, abarcando tipo de escritura, temas abordados y extensión de los poemas.

Liu, Fu y Cao [20] se enfocan al control del contenido retórico en los poemas, en este caso el control del vocabulario y la intención de las palabras, teniendo como meta, generar poemas con vocabulario que presenta dos tipos particulares de figuras retóricas, el paralelismo

[1] Aplicación multi propósito desarrollada en china, que ofrece desde mensajería, redes sociales, pagos online, etc. Link de Wechat: `https://www.wechat.com/es/`

(describir un objeto haciendo referencia a otro) y la humanización (dando propiedades humanas a objetos inanimados).
En un caso distinto a las redes neuronales Hämäläinen y Alnajjar [19] desarrollaron mediante algoritmos genéticos, un generador de poesía en idioma finés, adaptando los principales mecanismos como selección, cruzamiento y mutación para generar secuencias (versos). Para el control del vocabulario se usó Word2Vec pre entrenado en fines y los poemas se completan verso a verso hasta generar estrofas de semántica y extensión similar.
Lin, Gao y Chang [26] han usado los Transformers para la generación de historias, teniendo como base la generación de poesía, considerando las características propias del idioma y usando herramientas para la codificación del lenguaje.
Q. Wang, X. Wang, Liu y Chen [27], entrenaron un Transformer (BERT) para la predicción de la estructura prosódica (análisis del efecto fonético en las palabras) de un texto, en este caso un poema, a diferencia de artículos similares, donde se usan distintos modelos para identificar la estructura fonética a distintos niveles, ellos lograron que un solo modelo identifique estos niveles usando una clasificación multitarea, logrando aumentar el desempeño y reducir la cantidad de datos necesarios durante el entrenamiento.
Por último Köbis y Mossink [23] desarrollaron un estudio mostrando los avances de la generación de texto y cómo el texto artificial se acerca cada vez más al real, siendo en algunos casos indistinguible. Para eso se generaron poemas usando el Transformer GPT-2 y se elaboraron diversas comparaciones entre ellas, una con poemas de GPT-2 vs poemas de Maya Angelou (poeta americana muy reconocida), teniendo buena aceptación los poemas generados por el modelo.

Capítulo 3

Experimentos

Esta sección tiene como objetivo mostrar el proceso de desarrollo de los modelos de generación de poesía dividiéndose en dos fases. En la primera parte se usaron Redes Neuronales Recurrentes y en la segunda Transformers. También mostrarán los resultados obtenidos por los modelos.

3.1. Resultados primera fase: Redes Neuronales Recurrentes

En la primera fase se usaron redes recurrentes, en específico redes Gated Recurrent Unit (GRU) que son una variante de las recurrentes, la decisión fué tomada debido a las siguientes ventajas:

- Están diseñadas para evitar este problema común en las RNN, la explosión y desvanecimiento del gradiente.
- Necesita menos recursos computacionales: Primero solo usa dos (reset gate, update gate) compuertas a diferencia de LSTM que necesita 3 (forget gate, input gate, output gate). Asimismo las redes GRU no necesitan una unidad de memoria (como en las redes LSTM) para controlar el flujo de información.
- El anterior punto deriva en que las redes GRU tengan una velocidad de entrenamiento superior sobre las redes LSTM.
- Rendimiento similar a LSTM en corpus pequeños.

También se tomaron en cuenta las condiciones para realizar el experimento para seleccionar el modelo.

- Poder de cómputo limitado: para este caso solo se tenía como restricción y única solución para los experimentos el uso del equipo que provee Google Collaboratory de forma gratuita. Principalmente por los precios elevados para obtener equipos adecuados para ejecutar estos modelos, debido a que las redes recurrentes necesitan mucho poder de cómputo para entrenar (memoria RAM y VRAM).
- Tiempo de cómputo limitado: Google Collab tiene un límite de tiempo (no superior a 12 horas) por día, que varía en función de diversos factores como el porcentaje del uso del equipo o si estás usando una tarjeta gráfica. Debido a que Google no menciona cómo realiza el cálculo, el uso constante de los recursos deriva en menos horas disponibles a largo plazo.

Debido a lo anterior se considera que una GPU era la mejor opción a elegir entre los modelos disponibles.
La primera fase está dividida en dos secciones, la primera muestra los resultados de la generación de poesía con una codificación de las secuencias a nivel caracter y en la segunda la codificación es a nivel palabra.

3.1.1. Base de datos

La base de datos para la primera fase está escrita en un archivo de texto plano, donde para cada poema se escribe su título en mayúsculas y seguido el contenido del mismo, como se puede ver en la Figura 3.1.
La base de datos está formada por 400 poemas de un mismo autor, Pablo Neruda, la decisión fue tomada al considerar que es uno de los poetas en español de mayor reconocimiento a nivel mundial.

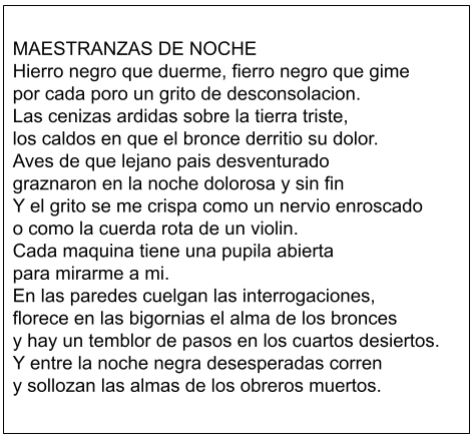
MAESTRANZAS DE NOCHE
Hierro negro que duerme, fierro negro que gime
por cada poro un grito de desconsolacion.
Las cenizas ardidas sobre la tierra triste,
los caldos en que el bronce derritio su dolor.
Aves de que lejano pais desventurado
graznaron en la noche dolorosa y sin fin
Y el grito se me crispa como un nervio enroscado
o como la cuerda rota de un violin.
Cada maquina tiene una pupila abierta
para mirarme a mi.
En las paredes cuelgan las interrogaciones,
florece en las bigornias el alma de los bronces
y hay un temblor de pasos en los cuartos desiertos.
Y entre la noche negra desesperadas corren
y sollozan las almas de los obreros muertos.

Figura 3.1: Forma de un poema en la base de datos

3.1.2. Proceso de entrenamiento de los 2 modelos

Para el entrenamiento de los modelos (generación nivel caracter y palabra) deben tomarse en cuenta los siguientes puntos:

- El entrenamiento se realizó en Google Collaboratory (se tienen límites de tiempo y de uso).
- El tamaño de la base de datos es de 400 poemas de un mismo autor.
- Uso de las librerías Tensor Flow y Numpy para el procesamiento de datos y la carga del modelo.

Los pasos necesarios para el entrenamiento de los modelos son los siguiente:

- Importar las librerías correspondientes para los datos y las redes.
- Cargar los datos a Google Colab, ya sea desde Google Drive o desde un equipo local.
- Abrir el archivo y remover puntuación (en este caso es remover caracteres desconocidos y dobles espacios en blanco).
- Determinar vocabulario y su tamaño: varía de acuerdo a si la codificación es por palabra o por caracter. Se obtienen todos los símbolos únicos.

- One Hot Encoding: Asignar un número distinto a cada elemento en el vocabulario para distinguirlo de los demás.
- Separar el texto por secuencias: Cada secuencia tiene una extensión determinada de elementos (ya sean palabras o letras).
- Generar token (caracter o palabra) objetivo para cada secuencia: En este caso es el token objetivo que se quiere predecir dado la secuencia construida en el paso anterior.
- Crear lotes de secuencias: Agrupar las secuencias en grupos de tamaño determinado.
- Asignar hiper parámetros: Son determinados por la persona y dependen de cada experimento. Entre ellos está el número de épocas, tasa de aprendizaje, dimensión del vector de características, etc.
- Construcción de la arquitectura de la red en capas (ver apéndice B): La capa de entrada es del embedding, las capas ocultas pertenecen a la red GRU y la capa de salida es una capa softmax.
- Entrenamiento del modelo y guardado del mismo al terminar. Google Colab puede guardar puntos específicos del entrenamiento para su posterior descarga, ya sea cada cierto número de épocas, el modelo con la mejor precisión o un número determinado de modelos para no saturar la memoria de Colab.

Una vez guardado, el modelo se puede cargar en Google Colab y generar texto. Cada modelo recibe como entrada una secuencia, ya sea de caracteres o de palabras.
Los valores que cambiaron en los experimentos fueron el número de épocas en el entrenamiento y la tasa de aprendizaje. el objetivo del entrenamiento fue buscar el modelo que produjera el menor número de errores léxicos y semánticos.
Además de considerar la forma del poema, en este caso, estar escrito preferentemente en verso debido a que es el formato común de los poemas. También es preferente que los versos tengan una extensión(esto es el número de sílabas que tiene cada versos) similar entre sí o que la diferencia de extensión entre cada versos no sea tan marcada para así considerar que tienen una mejor forma.
En cuanto a la comparación de los modelos por métricas como la función de perdida o la precisión, se descartó debido a que estos valores no reflejan la parte creativa de un texto, altos niveles de precisión no garantizan que el texto se parezca a un poema, aún así, cabe mencionar que la mayoría de los modelos presentados lograron precisión arriba de 0.90.

3.1.3. Resultados generación de poesía a nivel caracter

Condiciones iniciales de la red

Las condiciones iniciales para todos los experimentos fueron las siguientes:

- Tamaño del vocabulario = 57
- Tamaño total de caracteres en los datos = 401378
- Tamaño de los lotes = 64
- Número de unidades GRU = 512
- Longitud de la secuencia = 100
- Dimensión de embedding = 256

Experimento 1: 30 épocas y tasa de aprendizaje = 0.001

Para este experimento se utilizó una taza de aprendizaje de 0.001, en el caso de las épocas no se utilizó un número muy alto debido al reducido número de datos.
En este caso la salida, se puede ver en el Cuadro 3.1.
El modelo tiene las siguientes características:

- Genera texto con palabras inexistentes (que no están en los datos) como: Pasesia, Siguido, bienamad.
- Agrega a la entrada CASA los caracteres NO, en un intento por completar el título del poema.
- El formato del texto, esta en forma de verso.
- Se observa el uso de comas al final de la mayoría de los versos.
- Se puede identificar el uso de mayúsculas después del punto.

De acuerdo a la estructura del texto intenta describir un paisaje visto desde los ojos de una persona, menciona praderas y el mar, aunque el texto no se logra entender debido a la frecuente aparición de faltas de ortografía y palabras que no existen.

Entrada	Salida
CASA	CASANO Pasesia lo que fresca de tus ojos de relato. Siguido en desdichas de las praderas, en la inmensidad de mis bienas, todo estar en serio ahora. Despues el mar y recorriendo el oro. Muy bien, bienamad LR = 0.001

Cuadro 3.1: Resultado experimento 1 de generación por caracter

Experimento 2: 30 épocas y tasa de aprendizaje = 0.0015

Para este caso se incrementó en 0.0005 la tasa de aprendizaje con respecto al primer experimento y se usaron 30 épocas para el entrenamiento.
La salida mostrada en el Cuadro 3.2 tiene características parecidas a las mostradas en el primer experimento:

- Generación de palabras inexistentes como negolio, montoño y polvosiertos.
- Formato en verso, aunque los versos son más extensos que en el primer ejemplo, algunos renglones del texto podrían considerarse incluso párrafos y no versos debido a su gran extensión (un ejemplo de poemas con este tipo de verso es **Táctica y Estrategia** de Mario Benedetti) hay que señalar que había pocos ejemplos de este tipo de verso en los datos de entrada.
- No usa comas después de cada párrafo.

Sobre el contenido del texto, no es muy claro lo que quiere expresar, en un momento habla de agua, en otro de bodegas y al final caminos.

Entrada	Salida
CASA	El rey negolio sin agua solemne perdida que alcanzamos todo en un montoño de seda, tu y bodegas interminadas polvosiertos caminados o gastas caidos en el mismo frente a los caminos con mis hermanos n LR = 0.0015

Cuadro 3.2: Resultado experimento 2 de generación por caracter

Experimento 3: 30 épocas y tasa de aprendizaje = 0.005

Para el tercer experimento la tasa de aprendizaje se incrementó 5 veces con respecto al primer caso llegando a 0.005 y manteniendo la misma cantidad de épocas para el entrenamiento.
Como se ve en el Cuadro 3.3, el texto generado comparte las siguientes características y errores con los anteriores:

- El texto generado en este caso no respeta el salto al siguiente renglón luego de algún título.
- Tampoco pone mayúsculas en la siguiente palabra luego de un título como se puede ver en el texto para el caso de *reminanta* que debería ser *Reminanta*.
- Tiene más palabras inexistentes que los dos experimentos anteriores, como reminenta, inadar, Labera, algua, avertineces, etc.
- Llega a poner palabras que inician con mayúsculas (como si fueran nombres propios o de lugares).

El contenido es muy difícil de entender debido a todas las palabras inexistentes. Un aspecto interesante es que se forma texto en verso, de una extensión similar entre si, en comparación con los anteriores donde cada párrafo tenia una amplitud distinta. Se puede considerar un peor resultado que los dos anteriores en cuanto a contenido, aunque una mejora en cuanto a la forma, en este caso los verso tienen casi la misma extensión en sílabas, no como en los experimentos anteriores que unos versos eran mucho más largos que otros.

Entrada	Salida
CASA	CASA reminenta inadar la noche, alada a Labera, las algua de un acorto sila sus avestenices de arbol, de los meterios, incadas consiges, desunura, que volicios, en la cruzad mueza, el perro er la piquerte LR = 0.005

Cuadro 3.3: Resultado experimento 3 de generación por caracter

Experimento 4: 40 épocas y tasa de aprendizaje = 0.001

Con respecto al cuarto experimento, en lugar de cambiar la tasa de aprendizaje, se experimentó con el número de épocas, aumentando el número en 10 para el primer experimento y fijando la tasa de aprendizaje en 0.001.
El texto generado en este cuarto experimento se muestra en el Cuadro 3.4, donde se puede ver un comportamiento parecido a los primeros dos experimentos:

- Primero en la parte del título, el texto muestra que puede distinguir la entrada como un título, incluso añade más texto en mayúsculas para completarlo y en el siguiente renglón inicia con mayúsculas.
- Se observa una estructura de verso en todo el texto.
- Tiene palabras inexistentes como llomar, deseira y lentamines.
- En general no usa comas al final de los versos, no es obligatorio para la poesía, pero es usual encontrarlos en los poemas.

En cuanto al contenido del texto, se puede ver el intento de formar una idea, aunque aún no logra plasmarla. El texto de experimento tiene menos errores que en los anteriores modelos, pero todavía muestra la construcción de palabras desconocidas.

Entrada	Salida
CASA	CASA LA GUERRA Para que vuela en mi frente llomar al que quiso nacer y recibio el canto de mi deseira traes al relampago remoto los lentamines habitaciones, niñas durmiendo con las manos en el corazon, so LR=0.001

Cuadro 3.4: Resultado experimento 4 de generación por caracter

Experimento 5: 40 épocas y tasa de aprendizaje = 0.0015

El quinto experimento aumenta en 10 las épocas del entrenamiento usado para el experimento 2 dejando la cifra en 40 épocas totales y mantiene la misma tasa de aprendizaje del experimento 2 quedando en 0.0015 para el entrenamiento.
En cuanto al texto generado se puede ver en el cuadro 3.5, siendo una salida con características similares a los anteriores:

- Parece estar ordenado con una estructura en verso con todos los renglones teniendo una extensión similar.
- Aún se encuentran palabras inexistentes (ue, dejarmer, mueros, castilio).
- A diferencia del experimento 4 hace uso de más comas y respeta la mayúscula después de un punto.

Revisando el contenido del texto se puede ver que intenta completar una idea, incluso añadiendo la descripción de un entorno (París, pobres, castillos), siendo el modelo con mejores resultados hasta ahora, a pesar de los problemas que aún se observan.

Entrada	Salida
CASA	Es ue parte de Paris a dejarmer, a mi boca no te mueros, por pobres por todas partes. Hay el ciego sangriento. El hombre claro y sombrero, su castilio de olvido lejos de hilo, es que de llorar la llu LR = 0.0015

Cuadro 3.5: Resultado experimento 5 de generación por caracter

3.1.4. Resultados generación de poesía a nivel palabra

En este caso el vocabulario se forma por todas las palabras distintas que se encuentran en el texto del corpus (existe un mayor número de palabras que de caracteres), se usó un modelo similar al de caracter, solo que adaptado para trabajar con palabras. A diferencia de la generación por caracteres que se pasaba una palabra de longitud n, en este caso se le pasa una secuencia de palabras, una frase de tamaño n.

Condiciones iniciales

Las condiciones iniciales usadas para los experimentos son:

- Tamaño del vocabulario = 14855
- Tamaño total de palabras en los datos = 71574
- Tamaño de los lotes = 128
- Número de unidades GRU = 512
- Longitud de la secuencia = 10
- Dimensión del embedding = 256

Experimento 1: 30 épocas con tasa de aprendizaje: 0.001

Para el primer experimento se fijó 0.001 como tasa de aprendizaje y fué entrenado durante 30 épocas.
Para la parte del texto los resultados se pueden ver en el cuadro 3.6, se observan varios errores:

- Repetición de palabras varias veces: kilómetros, medidos, ESE, empuñadura.
- Uso de palabras en mayúsculas, correspondientes a títulos de poemas, en cualquier parte del texto, sin considerar si es un título o contenido del poema.
- No se observa la elaboración de ideas y parece más texto al azar.
- No se usan conectores entre las palabras: el, los, la, un, etc.

A pesar de eso se puede observar que el texto está en forma de verso y las extensiones son similares en casi todos.

Entrada	Salida
vida como por una aguja sangrienta entre las decisiones sin	ejercito, medidos, torpeza kilometros esta, medidos, mesas populacho comunicativa, ESE mineral ejercito, empuñadura, clima mineral comunicativa, yegua kilometros abanicos, puños ESE comunicativa, lavan, puños kilometros cuello empuñadura, sollozos. daños LR= 0.001 Precisión final =0.75104

Cuadro 3.6: Resultado experimento 1 de generación por palabras

Experimento 2: 30 épocas con tasa de aprendizaje: 0.002

Para el caso del segundo experimento se duplicó la tasa de aprendizaje subiendo de 0.001 a 0.002, manteniendo el número de épocas en 30 como en el experimento 1.
La salida generada por el experimento se puede observar en el Cuadro 3.7 donde el texto tiene las siguientes características:

- Renglones de distinta extensión, distinto al primero donde eran de la misma extensión.
- A diferencia del primer experimento no tiene el problema de palabras repetidas.
- No respeta reglas ortográficas como mayúscula después del punto.
- Muestra rastros de conectores como "son".
- Muestra signos de construcción de una idea aunque primitiva, en este caso relacionada con el agua ya que usa palabras como navío, oxígeno, buque lluvioso, Emerge.

Entrada	Salida
de un violin. Cada maquina tiene una pupila abierta para	ronda. navio, tiendo di oxigeno muerte flechazos tierno recuerde buque mi. mañanas. ciegos, prolongado mirad costados lluvioso truncado cima Emerge Iban golpeaba son caciques, corazon abrete cualquiera, cantado, truncado comunicativa, LR= 0.002 Precisión final = 0.94539

Cuadro 3.7: Resultado experimento 2 de generación por palabras

Experimento 3: 30 épocas con tasa de aprendizaje: 0.0015

En el tercer experimento se aumentó en 0.005 la tasa de aprendizaje con respecto al primer experimento, manteniendo las mismas 30 épocas para el entrenamiento.
En relación al texto generado mostrado en el Cuadro 3.8, el resultado tiene errores que se encontraron el primer experimento, pero también rasgos parecidos al del segundo:

- Se encuentran palabras repetidas, en este caso "kilometros oceanicos" aparece dos veces en el mismo verso,"sostenidos dias" están en versos consecutivos, maquinista también, pero está muy separada.
- Una palabra totalmente en mayúsculas en medio del poema.
- Los versos son de muy distintas extensiones entre ellos, hay algunos cortos agrupados en medio y otros muy grandes en las esquinas.
- Tiene varias faltas de ortografía, como no poner mayúscula después de un punto.
- En cuanto al contenido del texto, la idea otra vez es relacionada al océano, en el caso de la entrada, tiene la palabra "mar" incluida. Se puede interpretar como un avance al poder relacionar la entrada con lo que construye el modelo.

Entrada	Salida
tan pequeño Pequeño frente al horizonte y frente al mar	pueblan espaciosas maquinistas, kilometros oceanicos. cejas. verde kilometros oceanicos. agrupa honor horario. sostenidos dia, cielo contorno sostenidos dia, navegaramos Stalingrado CIELO vulgar daños señorita daños vulgar ejercito, amigo silbidos, maquinistas, LE= 0.0015 Precisión final= 0.93415

Cuadro 3.8: Resultado experimento 3 de generación por palabras

Experimento 4: 50 épocas con tasa de aprendizaje: 0.0012

Para el cuarto experimento, se usó una tasa de aprendizaje de 0.0012 y 50 épocas para su entrenamiento siendo ambas cifras mayores a las fijadas en el experimento 1 donde la tasa de aprendizaje era 0.001 y las épocas 30.
Esta vez la salida mostrada en el Cuadro 3.9 fué más desordenada que en las anteriores mostrando un comportamiento inusual:

- No ordena el texto en versos, sino como texto en prosa ,existe poesía en prosa, aunque no tan usada. La única característica constante en todos los anteriores modelos, la característica de texto en verso también se presentó en los modelos de generación por caracter.
- Parece un listado de palabras que no se relacionan más que tener apariencia de poema.

- Se repite la palabra "vulgar".
- Existe una palabra en mayúscula a medio texto.
- No se visualiza una idea clara en el texto.

Entrada	Salida
en la consumación ferviente de la olla, y el jiron	medidos, establezco, escojo dejo, frescos, establezco, vulgar pasmoso, olvido, vetas, alas. kilometros has embarcaciones alas. vulgar Eran frailes daños Zuñiga, empuñadura, PRIMER pajarillo Zuñiga, vulgar soñolientamente, escupitajo sobrante, Yaco vulgar LR= 0.0012 Precisión final = 0.9647

Cuadro 3.9: Resultado experimento 4 de generación por palabras

Experimento 5: 50 épocas con tasa de aprendizaje: 0.00115

Para el quinto experimento, la tasa de aprendizaje se estableció en 0.00115 (un leve aumento en comparación al primer experimento que tenía 0.001) y el número de épocas se fijó en 50.
En cuanto al contenido que generó el modelo mostrado en el Cuadro 3.10, presenta características parecidas al anterior experimento desde la ortografía, hasta la forma:

- El texto está en prosa y no en verso, siendo el segundo modelo en generar este tipo de texto.
- La palabra "vulgar" sigue presente en gran parte del texto, asimismo aparece 2 veces "Wiracocha" (la palabra aparece una sola vez en el corpus).
- Errores ortográficos: No hay mayúscula después del punto.
- El texto en general no tiene una idea de que quiere expresar.

Entrada	Salida
despues de declararlo inutil y anacronico camello, cuando por largos	daños noche Rio, puños viene. diaria vulgar mañanas. turquesas, Wiracocha, estrellita muertes, daños fueran vulgar vulgar agrupa aji, caballerias llueven ejercito, Wiracocha, crepusculares vulgar vulgar otro vulgar Cuida habitos gorilas, LR = 0.00115 Precisión final = 0.9621

Cuadro 3.10: Resultado experimento 5 de generación por palabras

Experimento 6: 50 épocas con tasa de aprendizaje: 0.00125

En este último experimento, la tasa de aprendizaje se estableció en 0.00125 siendo un poco mayor al del quinto experimento. El número épocas de entrenamiento se fijó en 50

épocas.
En este caso el contenido mostrado en el Cuadro 3.11, el texto no tiene relación alguna cuando se examina el poema original de donde fue extraída la frase.

- El texto esta vez está en forma de verso a diferencia de los dos experimentos anteriores.
- Sigue siendo más parecido a una lista de palabras que un poema. La mayoría de las palabras no tienen relación cercana entre sí.
- Prevalece el error ortográfico de no asignar una palabra con mayúscula después de un punto.
- Sobre el contenido del texto, el poema no tiene sentido.

Entrada	Salida
darme lo que tienes a la tierra viniste como yo	ropa interminable hacendados. algun hacendados. armados creiamos anduvieron, un digital, amontonados atardeciendo, dejo, Isaacs, mano, arrullo. puños viajar caballeros golpea, simples tierno laringe nuestra y, daños vulgar anduvieron, corales lagartija LR=0.00125 Precisión final = 0.96269

Cuadro 3.11: Resultado experimento 6 de generación por palabras

3.2. Resultados Segunda fase: Transformers

3.2.1. Base de datos y Modelo

El primer paso para generar poesía en español con Transformers es tener una base de datos de poemas en español. El formato de la base en este caso es de tipo JSON, formato muy usado debido a que es muy descriptivo y fácil de entender, en este caso para cada poema se incluyen el título y contenido. Para esto tomando como punto de inicio la base de datos de la primera fase se recopilaron 10 mil poemas en español entre autores latinoamericanos y españoles, asimismo en menor número poemas traducidos al español. El peso total aproximado de la base de datos es de 10MB. Para los experimentos se decidió dividir la base de datos en tres subconjuntos

- Primer subconjunto: 5100 poemas del total.
- Segundo subconjunto: 7500 poemas del total.
- Tercer subconjunto: Los 10000 poemas.

También se decidió utilizar 85 % del total de datos de los subconjuntos para entrenamiento y 15 % para prueba.
Para la selección de los modelos se consideraron dos requisitos:

- El primer requisito que deben cumplir es ser modelos pre entrenados, es mejor usar este tipo de modelos debido al alto costo computacional y el tiempo requerido para entrenar un Transformer desde cero.
- El segundo requisito es que el modelo seleccionado debe de estar pre entrenado en idioma español.

El primer problema se soluciona usando los modelos que provee **Huggingface** [1]. La ventaja de los modelos pre entrenados es que se pueden refinar (fine tuned) con una cantidad pequeña de datos para realizar tareas específicas, esto en comparación con la cantidad necesaria para entrenarlo desde cero.
Para el segundo problema, de entre los modelos disponibles el seleccionado fué GPT-2 el cual fué desarrollado en 2019, es uno de los modelos más famosos en la actualidad, teniendo el modelo original distintas versiones, principalmente se divide de acuerdo al número de parámetros que tiene, los parámetros se entienden como las variables que se irán modificando y ajustando de forma automática durante el entrenamiento, entre las que están los peso de la red, ver apéndice B. Va desde su versión más pequeña (small) de 124 millones de parámetros, la versión mediana (medium) de 345 millones de parámetros, la versión grande (large) de 762 millones de parámetros y la versión extra grande de 1542 millones de parámetros.
Originalmente fue entrenado en una base de datos conocida como WebText formada de 45 millones de páginas web en inglés, seleccionadas de acuerdo a su relevancia y el peso final de la base de datos es de 40 GB que no está liberada al público.
Además, GPT-2 tiene una versión en español, *GPT2-small-spanish* que fué entrenado y evaluado por Datificate, el cual está basado en una versión portuguesa de GPT-2 [24], ambos toman como base la versión más pequeña de GPT-2 de 124 millones de parámetros. *GPT2-small-spanish* ha sido entrenado en una base datos correspondiente a todos los datos en español contenido en Wikipedia en español pesando alrededor de 3 GB.
Otra razón para la selección de GPT-2 es porque su propósito original es la generación de texto, a diferencia de otros que están enfocados en resolver otras tareas de Procesamiento de Lenguaje Natural como traducciones, clasificación, resumen de textos, etc. Además, el vocabulario que manejan los modelos asciende a 50000 palabras distintas abarcando gran parte del vocabulario total en español, 93 mil palabras e inglés, 70 mil palabras.
Finalmente, el entrenamiento de estos modelos con los poemas en español se llevó a cabo en Google Collaboratory usando una Nvidia Tesla K80 con 12GB de VRAM, con procesador Intel(R) Xeon(R) a 2.30GHz, con 12GB de RAM y 20GB de almacenamiento.

3.2.2. Proceso de entrenamiento de los modelos

Puntos tomados en cuenta para el entrenamiento de los modelos Transformer.

- Todo el entrenamiento toma lugar usando el servicio Google Collab, teniendo tiempo y poder de cómputo limitados.
- La base de datos completa es de 10 mil poemas.
- Se usó la librería Hugging Face que proporciona modelos Transformer pre entrenados.

Los pasos necesarios para el entrenamiento de los modelos son los siguientes:

- Descargar la librería de Hugging Face e importarla.
- Descargar la base de datos, ya sea desde Google Drive o desde el equipo local.

[1] https://huggingface.co/

- Descargar la versión del modelo GPT-2 a usar. En este caso para los experimentos se usaron GPT-2 en español y GPT-2 en inglés en dos versiones, la pequeña y la mediana.
- Descargar el tokenizador correspondiente: debe ser el tokenizador del modelo, otro puede estar en otro idioma y tener otro vocabulario. El tokenizador de GPT-2 genera una codificación a nivel sub palabra usando Byte Pair Encoding (ver apéndice C).
- Obtener el subconjunto de poemas a usar.
- Obtener el contenido de los poemas.
- Dividir los datos entre datos de prueba y datos de entrenamiento.
- Tokenizar los datos (pasar de texto a número).
- Determinar el tamaño de los bloques para cada época (determina cuantas secuencias tendrá cada bloque).
- Asignar los argumentos de entrenamiento (hiper parámetros).
- Entrenar y guardar el modelo.

Los modelos GPT-2 son muy pesados en comparación a otros modelos de redes. La versión pequeña pesa 500 MB y la versión mediana 1.35 GB. Así que descargar un modelo desde Google Colab hacia el escritorio tarda mucho tiempo y es mejor guardarlos en Google Drive al final del entrenamiento del modelo.
Al cargar el modelo, este recibe como entrada secuencias de datos codificadas, por esto hay que descargar el tokenizador correspondiente para los datos.

3.2.3. Experimentos y resultados

Copias de los modelos originales fueron entrenadas bajo distintas condiciones para analizar las mejoras en los poemas (contenido, forma) de acuerdo al nivel de entrenamiento. También fueron entrenados bajo distintos tamaños de la base de datos, para ver el impacto de los datos en el poema.
Además, se realizó un cuarto experimento el cual consiste en entrenar versiones de GPT-2 original en inglés (medium y small) con la base de poemas completa y verificar los resultados.
El modelo recibe una entrada (fragmento de un poema) y devuelve un poema generado automáticamente con una extensión variable (máximo de 100 tokens).

Experimento 1

En el primer caso se empleó el modelo GPT-2 en español y una base de datos con solo 5100 poemas tomados al azar de la base de datos completa. Siendo 85 % (4335) para el entrenamiento y 15 % (765) para prueba.
El modelo ya entrenado generó las salidas mostradas en los Cuadros 3.12−3.14. Cabe señalar que cada poema se generó en un intervalo de elementos fijo, pero se recortó en algunos casos el final debido a que el verso quedaba a la mitad. Se dejó hasta donde se completará el último verso o decidiera terminar el modelo. Por eso los poemas tienen distintas extensiones.
En el Cuadro 3.12, se muestra un poema dividido en versos de tamaños parecidos. En los primeros dos versos maneja dos conceptos *corazón* y *alma* los cuales retoma en el tercer verso, mostrando su capacidad para mantener la relación entre los elementos que generó anteriormente.

Aunque no usa algunos signos de puntuación comunes que se pueden ver en versos generados por humanos, como poner una coma después de cada verso, aunque no ocurre en todos los poemas.

Entrada	Salida
Te hablo Luna te hablo Sol quisiera ser la llave de tu corazón.	Tu corazón es la puerta de la noche, y tu alma es el camino de los muertos que se abren en la luz de las tinieblas como un relámpago de luz que cae sobre la tierra para que el sol se apague y caiga sobre el mundo con un beso de luna y un abrazo de sol. Tú eres mi alma, tu cuerpo, mi cuerpo.

Cuadro 3.12: Poema 1 Experimento 1 de generación con Transformers

En el Cuadro 3.13 la entrada proporcionada al modelo tenía muchos signos de admiración para demostrar efusividad, en este caso el poema los replicó dando así a entender que mantenía la misma energía que los versos original.
Claramente se observa el uso de mayúsculas en los sujetos como en el poema original, además de mantener una idea central, *Dios*, para todo el poema.

Entrada	Salida
Hoy cantan los cielos la Gloria de Dios, que alaben los hombres.	Oh Dios!, ¡oh Gloria!, que en la gloria de la vida, la Gloria del Padre, y el Hijo del Hijo, se encienden al cielo, a la tierra y al mar! ¿Quién es Dios? Yo no sé, pero sé que Dios es un hombre,

Cuadro 3.13: Poema 2 Experimento 1 de generación con Transformers

Para el último poema, Cuadro 3.14, el modelo recibió de entrada un poema en coplas, en este caso logró replicar la forma y el contenido está relacionado entre sí. Dando a entender que logró aprender la forma del poema original. Siendo este un punto relevante, ya que los anteriores poemas generados también han logrado imitar la métrica de sus poemas de entrada. Otro detalle interesante es que comienza una palabra con mayúsculas después de un punto.

Entrada	Salida
Mi árbol tenía sus ramas de oro. Un viento envidioso robó mi tesoro. Hoy no tiene ramas. Hoy no tiene sueños.	Hoy no tengo hambre. Hoy es el día de la muerte. No hay nadie que viva, ni nadie en la tierra, ni en el mar. Sólo el agua, el viento y el fuego. La muerte es un sueño. El mar es una muerte que no existe. Es un mar sin vida.

Cuadro 3.14: Poema 3 Experimento 1 de generación con Transformers

Experimento 2

En el segundo experimento se usó el subconjunto de la base de datos original que contiene 7511 poemas. con igual distribución que el anterior, 85 % (6384) para entrenamiento, 15 % (1127) para prueba.
En este caso, el entrenamiento cambió en el aumento de los datos, para el caso del Cuadro 3.15, se muestra uno de los poemas generados por el nuevo modelo, se observa el uso de elementos retóricos en el poema para describir a una persona, que es una idea completamente distinta a la presentada en el Cuadro 3.12, lo cual muestra el impacto inmediato del entrenamiento al generar poemas más variados.
En el Cuadro 3.16, se puede observar que el poema generado se aleja de la idea original

Entrada	Salida
Te hablo Luna te hablo Sol quisiera ser la llave de tu corazón.	Tu cuerpo es el espejo de la vida, el reflejo de tus ojos y la luz de tu boca. Tu boca es un espejo que se abre y se cierra para que la noche se abra como un túnel de luz. Tus manos se abren para ver el mundo mientras que el sol se apaga en el fondo del espejo.

Cuadro 3.15: Poema 1 Experimento 2 de generación con Transformers

del texto de entrada, genera un poema más cercano a una plegaria mientras narra una historia, siendo un poema que en lugar de buscar la rima, prefiere transmitir un mensaje. Se puede ver la complejidad de la idea armada, yendo más allá del poema original. Por último, en el Cuadro 3.17 se incluye el resultado de darle como entrada unas coplas, no conserva la forma exacta del poema original, aunque sí conserva rasgos, a cambio, la idea es más completa que en el Cuadro 3.14.
Por lo que se ha visto en este experimento, el detalle en el contenido de los poemas es mayor al del primer experimento, y aún así, conservan los rasgos de forma de la obra original. El único motivo del cambio es el impacto de un mayor número de poemas durante

Entrada	Salida
Hoy cantan los cielos la Gloria de Dios, que alaben los hombres.	¡Oh, Dios mío, cuántas veces he vivido en esta tierra! Y cuántos años he estado aquí, en este país? Yo no sé si alguna vez ha pasado, pero sé que estoy aquí y estoy triste y he perdido la vida por el camino que lleva mi vida y mi muerte a la tierra de los muertos!...

Cuadro 3.16: Poema 2 Experimento 2 de generación con Transformers

el entrenamiento.

Entrada	Salida
Mi árbol tenía sus ramas de oro. Un viento envidioso robó mi tesoro. Hoy no tiene ramas. Hoy no tiene sueños.	Hoy no tengo tiempo. Hoy sólo tengo un sueño que es un recuerdo de mi infancia y que no es el recuerdo de mis años de juventud ni del amor que he tenido con la muerte que me he dejado en la memoria de todos los días, mañana no ha de ser mañana,

Cuadro 3.17: Poema 3 Experimento 2 de generación con Transformers

Experimento 3

Finalmente, para el tercer experimento se usó la base completa con 10 mil poemas divididos con 85 % (8512 poemas) del total para entrenamiento y 15 % (1503) para prueba. Los resultados en este caso fueron muy distintos a los esperados, en cada uno de los poemas generados, la idea del poema original se usa sólo de puente para desarrollar ideas más profundas y complejas que en los anteriores dos, siendo una tendencia para todos los poemas generados en este experimento, olvidando la forma original y centrándose completamente en el contenido del poema, esto hace que el poema se sienta más original que en los experimentos anteriores.
En el primer poema, Cuadro 3.18, el vocabulario es más envolvente y misterioso que en los dos anteriores, pero se puede ver que la forma del poema no está balanceada, aunque conserva su propiedad de versos.

Entrada	Salida
Te hablo Luna te hablo Sol quisiera ser la llave de tu corazón.	En las noches de lluvia y frío, en las mañanas de frío y caliente, cuando el viento sopla y el cristal de la ventana brilla, me parece que te miro y te veo en la alcoba oscura y sola. Quédate aquí, noche tras noche, sola en tu almohada, con tu sombra de mi sombra fiel, sin que nadie te vea.

Cuadro 3.18: Poema 1 Experimento 3 de generación con Transformers

Para el poema del Cuadro 3.19, existen faltas de ortográficas que no existían en los anteriores experimentos, al referirse a algunos objetos, más específico en el verso, *"Y las estrellas se apaga"*, donde se refiere a estrellas en plural pero *apaga* lo maneja en singular. Como se dijo antes, solo toma el poema inicial como punto de apoyo, pero no maneja mucho del vocabulario que presenta el original.

Entrada	Salida
Hoy cantan los cielos la Gloria de Dios, que alaben los hombres.	¡Oh, bienaventurado Albergue a cualquier hora! A la hora del alba, cuando el sol se extingue Y las estrellas se apaga, y el mundo se desvanece bajo el silencio de la noche, Cuando la tierra se convierte en cielo, y el cielo en mar.

Cuadro 3.19: Poema 2 Experimento 3 de generación con Transformers

Por último, el poema del Cuadro 3.20 es el mejor ejemplo de profundidad en una idea, ya que la idea de árbol la lleva más allá, introduciendo a un personaje en primera persona y comparando su existencia con ese árbol, y generalizando más adelante, aunque es más corto que los mostrados en los Cuadros 3.14 y 3.17, además de no contar con la forma del original. El poema es más maduro que los dos anteriores en cuanto al desarrollo de ideas complejas.

Entrada	Salida
Mi árbol tenía sus ramas de oro. Un viento envidioso robó mi tesoro. Hoy no tiene ramas. Hoy no tiene sueños.	Yo no sé Cuánto me he mirado Hoy no hay más que un árbol. Yo sé cuántos árboles hay en el mundo. De todos los que no he visto No sé dónde están los nombres de mis árboles

Cuadro 3.20: Poema 3 Experimento 3 de generación con Transformers

Experimento 4

Para el experimento 4 se entrenó la versión de GPT-2 original en inglés de tamaño mediano (de 345 millones de parámetros) con los 10 mil poemas en español.
El objetivo es observar la capacidad de los modelos Transformers para adaptarse a tareas en otros lenguajes para los que no estan entrenados específicamente y ver qué logra.
Para el primer poema ver Cuadro 3.21, se puede notar el uso del vocabulario dado como entrada, además de tener sentido cada verso dentro de la estrofa. Se puede ver también el uso correcto de los signos de admiración e interrogación. La única falla notable es el poco, o nulo para este ejemplo uso de comas al terminar un verso.

Entrada	Salida
Te hablo Luna te hablo Sol quisiera ser la llave de tu corazón.	¿Quién ha sido? ¡Oh, qué dulce amor! No puede ser más que la muerte, que en la noche de la luna no se siente la vida ni la tierra, ni el viento sino el cielo y el agua y los ojos y las aguas de una mujer que no tiene nadie porque no sé si espera nada

Cuadro 3.21: Poema 1 Experimento 4 de generación con Transformers

En el caso del poema en el Cuadro 3.22, el comportamiento es similar en el uso de las comas, a pesar de eso el poema que está generando se puede considerar bueno, ya que la lógica es buena y la estructura, aunque no parecida a la entrada, sigue generando versos.

Entrada	Salida
Hoy cantan los cielos la Gloria de Dios, que alaben los hombres.	¿Quién habla? ¡Qué dulce amor! La Gloria, la Gloria del alma, que al abrirse los ojos de la tierra y el viento y la luz de los árboles se deslizan en la noche como una lámpara de oro en el agua de un pájaro y las aguas donde el sol se parecen en las noches

Cuadro 3.22: Poema 2 Experimento 4 de generación con Transformers

Para el poema del Cuadro 3.23, las comas aparecen en varios ejemplos con esta entrada, además se puede notar como solo usa los signos de admiración al final, aunque los de interrogación si están al inicio y al final de una pregunta. También se observa que la idea a pesar de tener coherencia no está tan conectada a la entrada como las anteriores salidas de otros experimentos.

Entrada	Salida
Mi árbol tenía sus ramas de oro. Un viento envidioso robó mi tesoro. Hoy no tiene ramas. Hoy no tiene sueños.	Hijo, hijo mío! ¿Quién me habló? No sé qué dónde está en la noche, ni en el cielo de la lluvia, ni inmóvil! Déjame, Dios, que no te quiero en mi vida y en mis ojos, porque no me quieres

Cuadro 3.23: Poema 3 Experimento 4 de generación con Transformers

La causa de estos defectos en los tres poemas, es que, a pesar del entrenamiento al modelo aún le cuesta reconocer todas las reglas gramaticales usadas en el español. Es importante recordar que el modelo original fué entrenado solamente en inglés, además de la poca cantidad de poemas que se le proporcionó para lograr un aprendizaje total de las reglas, a diferencia de la versión en español.

Experimento 5

En este último experimento se usó la versión pequeña de GPT-2 (de 125 millones de parámetros, en la que se basa el GPT-2 en español) pero en inglés.
A diferencia de la versión medium, en este caso se observa desde el primer poema generado un mayor uso de signos de puntuación y admiración, como en el Cuadro 3.24 donde incluso usa punto y coma (;), asimismo respeta las reglas para los signos de admiración e interrogación.
Para el segundo poema del Cuadro 3.25, en este caso la forma del poema generado varía

Entrada	Salida
Te hablo Luna te hablo Sol quisiera ser la llave de tu corazón.	¿Quién te acuerdas de la noche y de tus ojos? Porque tú no sabes que te quiero porque no te olvides. ¡Oh, qué dulce esperanza, cuando te amas, cuando ames! Aquí no hay más que un día para ti, para mí, una mujer; tú, tu amor y yo, mi amada.

Cuadro 3.24: Poema 1 Experimento 5 de generación con Transformers

claramente en la extensión de cada verso sin que esto le quite lógica en general. Aquí no aparecen muchas comas al final de los versos, un elemento a denotar es la capacidad de usar **qué** para una oración en admiración y no solo para preguntar, esta es una regla que, se supone aprendió durante el entrenamiento.
El último poema en el Cuadro 3.26, la forma de abordar la idea del poema de entrada es más sombría que en modelos anteriores, además de la idea de deshumanizar se hace presente, quitando varios elementos propios del ser humano solo dejando al hombre vacío, está bien desarrollada en pocas líneas, algo que no logró de tan buena forma el modelo en el experimento 4.

Entrada	Salida
Hoy cantan los cielos la Gloria de Dios, que alaben los hombres.	¡Ay, qué tristeza de la Gloria! Porque en la tierra no hay nadie ni la Tierra. ¿Quién ha dicho que esta mañana se ha dejado en el fondo de la noche? No sé si alguna vez les ha pasado aún más que una mujer y que la muerte y el alma se han ido como un árbol de piedra y de hueso.

Cuadro 3.25: Poema 2 Experimento 5 de generación con Transformers

Entrada	Salida
Mi árbol tenía sus ramas de oro. Un viento envidioso robó mi tesoro. Hoy no tiene ramas. Hoy no tiene sueños.	¿Quién ha sido así? ¿De dónde estará mi corazón, mi amor, mi alma y mi espíritu y mi cuerpo, la tierra y la lluvia en el fondo de mi vida? ¡Ay, qué miedo! Porque no hay más que un hombre

Cuadro 3.26: Poema 3 Experimento 5 de generación con Transformers

Se debe señalar que los idiomas inglés y español tienen distintas reglas gramaticales, así que estos errores pueden deberse a que aún no ha aprendido de forma correcta las reglas del idioma nuevo. A pesar de eso en general el vocabulario lo aprendió exitosamente.

3.3. Encuesta

3.3.1. Descripción

El objetivo de esta encuesta es conocer la opinión de las personas sobre los poemas generados por un modelo transformer de forma automática. No se busca saber si los poemas tienen calidad o características específicas, solo se indaga si a las personas les gustan los poemas artificiales. El formulario utilizado se puede encontrar en el apéndice F.
Y se tienen dos preguntas fundamentales en el cuestionario

¿Te gusta el siguiente poema?

El objetivo es conocer si al encuestado le gusta un poema que se muestra anexado a la pregunta. El poema está en español y es generado por una máquina.

¿Qué poema te gusta más?

Para el segundo caso la pregunta presenta dos poemas que son similares en cuanto a la forma (tiene versos con una métrica parecida), el tamaño del poema (el número de renglones) y el tema que aborda el poema. Además, uno de los poetas está generado por una máquina y otro fue creado por un poeta profesional. Se quiere conocer qué poema prefiere, le gusta o le agrada el encuestado considerando que son poemas parecidos.

3.3.2. Universo muestral

El universo muestral consiste en personas de la red social Facebook, teniendo un rango de edad entre 18 y 70 años. Asimismo, los individuos no tienen estudios relacionados a poesía, siendo inexistente un conocimiento profesional sobre las características y diversidad de la poesía.

3.3.3. Medio Utilizado

Considerando la situación de la pandemia de Covid-19 en México, la decisión fué utilizar medios digitales, es decir, redes sociales y correo electrónico para distribuir y aplicar la encuesta.
Google Form fué seleccionada para la elaboración del formulario, debido a lo amigable de su interfaz a la hora de generar y editar preguntas de distintos tipos, además de permitir guardar los resultados en Google Drive de manera automática.
Otro beneficio de Form es la capacidad de generar ligas para la distribución de la encuesta por distintos medios. Cada persona encuestada recibió un link al formulario el cual guarda las respuestas al finalizar las preguntas.

3.3.4. Métrica

En este caso la métrica usada es el gusto o agrado de las personas hacia el texto que se les presenta, o en su caso qué texto les gusta más entre las opciones disponibles.

3.3.5. Resultados de las encuestas

Pregunta 1¿Te gusta el siguiente poema?

Resultados: 74.7 % SI, 25.3 % NO
Los resultados muestran que al 77.2 % de los encuestados les gusta el poema generado por la máquina y al 22.8 % no les agrada. Para este caso el poema es un texto muy fácil de entender que combina versos de arte mayor y de arte menor, en poesía los versos se pueden clasificar de acuerdo al número de sílabas de los que están compuestos, un verso de arte menor está compuesto por 8 sílabas o menos y un verso de arte mayor está compuesto por 9 sílabas o más, ver apéndice E, siendo un poema con versos no uniformes. Además, el tema central es la vida usando palabras relacionadas al tema cómo vida, corazón y alma. Hay que señalar que ninguno de los versos rima con otro.

Pregunta 3 ¿Te gusta el siguiente poema?

Resultados: 61.4 % SI, 38.6 % NO
Los resultados muestran que al 45.6 % de las personas encuestadas no les gusto el poema (generado por la máquina), en cambio, al otro 54.6 % si les agrado el poema artificial.
El poema es un texto muy fácil de entender y está compuesto en general de versos de arte mayor y el tema principal es la vida donde se puede ver palabras relacionadas al tema como son alma, cielo y tierra, se puede ver que las figuras retóricas están aplicadas en todo el texto. También se debe señalar que ninguno de los versos rima con otro.

Pregunta 2 ¿Qué poema te gusta más?

Resultados: 75.9 % Opción 1, 24.1 % Opción 2
Acorde a los resultados se observa que el 78.9 % de las personas prefirieron el poema artificial donde el tema principal es la vida acentuado por palabras que representan objetos agradables a la vista como beso, flor, estrella. Se caracteriza por tener una extensión de

arte mayor en la mayoría de sus versos, además de ser fácil de leer. En cambio el segundo poema creado por el humano y que habla de un soldado solo recibió el 21.1 % de los votos totales, ambos poemas coinciden en no tener rimas, además los versos varían en extensión, entre arte mayor y arte menor. La diferencia fundamental es la forma de abordar el tema, en un caso usa el vocabulario para preguntarse lo que es la vida y en el otro busca atacar lo que quiere describir usando un efecto más agresivo.

Pregunta 4¿Qué poema te gusta más?

Resultados: 45.8 % Opción 1, 54.2 % Opción 2
Conforme a los resultados se ha obtenido que poco más de la mitad de los encuestados prefieren el poema creado por el humano (52.6 %) contra el poema artificial (47.4 %).
Las personas prefirieron un poema donde todos sus versos son de arte menor, donde se habla de la guitarra usando diversos recursos literarios como humanización. El poema artificial es de arte menor al igual que el humano y el tema central es la tierra y es más extenso que el humano al menos en versos. Ambos poemas son similares en extensión y estilo y el vocabulario no representa un polo opuesto entre sí, debido a que los temas no son opuestos y ambos tienen cómo objetivo la descripción de objetos inanimados.

Capítulo 4

Conclusiones

En primer lugar, hay que resaltar las principales aportaciones que son el desarrollo de modelos de redes neuronales, capaces de generar poesía en español. Por ejemplo, no hay antecedentes registrados de uso de Transformers para esta tarea específica siendo este uno de los primeros trabajos que usan estos modelos para generar contenido creativo y que sea agradable al público. De hecho, la sola búsqueda de bases de datos de poesía de un tamaño aceptable para los entrenamientos fue infructuosa encontrando bases de poemas pequeñas, debido a esto también se puede considerar como una aportación la base de datos creada con más de 10 mil poemas en español.
Para las conclusiones se deben separar las redes recurrentes y los Transformers debido a que los resultados mostraron características muy diferentes. Además, se deben señalar los resultados de las encuestas y como muestran qué tan agradables son los poemas generados por el modelo Transformer. Por esto las conclusiones están divididas en tres partes principales la primera está centrada en las redes neuronales recurrentes, la segunda habla de los Transformers y por último la tercera parte está dirigida a los resultados de las encuestas.

Sobre los modelos de redes neuronales recurrentes se puede concluir lo siguiente:

- De acuerdo a los resultados mostrados, el contenido generado por las redes neuronales recurrentes, considerando ambos, la generación por caracter y palabra, no se puede considerar poesía, debido a que, en su mayoría, los textos no mostraron ningún desarrollo de ideas sobre algún tema, además, de tener muchos errores ortográficos en cada texto.

- En el caso de la generación por caracter, los principales errores venían de la generación de palabras inexistentes, que afectan a la lógica general del texto impidiendo expresar las ideas claramente. Otro factor son los errores ortográficos como combinar mayúsculas y minúsculas, no respetar los puntos o los signos de admiración y exclamación, etc. Un factor a destacar es que en la mayoría de los casos logró generar texto en verso.

- Lo errores durante la generación por palabras son principalmente sobre el contenido del texto, el texto generado no tenía sentido, normalmente no hablaba de nada, ni se reflejaba ningún tema al conectar las palabras en los versos, además de ser en algunos casos un listado de palabras, también se debe señalar que sufría del uso repetitivo de palabras, como “vulgar”, encontrándose ésta más de una vez en el mismo renglón. En la parte de la forma, varios modelos dejaron de producir contenido en verso y generaron prosa.

Sobre los modelos de los Transformers se puede concluir los siguiente:

- De parte de los Transformers, el texto generado en la mayoría de los casos se puede considerar texto poético, lo cual es un avance enorme en comparación con las redes

recurrentes, donde se puede destacar el uso de gran parte de reglas gramaticales, como el uso de puntos, comas, signos de admiración e interrogación, Mayúsculas después de un punto, acentos, etc.

- En todos los casos generaron texto en verso y en la mayoría de los casos la métrica de los versos era cercana a la que se tenía en el texto de entrada. Siendo una muestra de la potencia para aprender estructuras de poemas, a pesar de no lograr siempre que todos los versos tuvieran la misma métrica.

- El poema generado por el modelo va acorde con el tema que maneja en la entrada, tomando las ideas y modelándolas para generar poemas relacionados usando vocabulario relacionado a la idea central. Y en los mejores casos, los poemas mostraron rasgos de desarrollo de ideas profundas, haciendo uso de diversos elementos literarios como humanización, como **el sonriente sol o la cartera hambrienta**.

- La versión en inglés del modelo y que fué entrenada con la base de datos en español logró generar poemas en español que se pueden considerar aceptables a pesar de algunos errores ortográficos, claro, tomando en cuenta el cambio de reglas gramaticales entre dos idiomas y que la base de datos era muy pequeña en comparación con las que se suelen usar para entrenar Transformers para que aprendan un idioma por completo.

- No es adecuado comparar los modelos Transformers con las redes neuronales mediante el uso de métricas como pueden ser comparación de la precisión, debido a que los valores altos o bajos de éstas no representan que tan bueno será el texto generado.

Sobre los resultados de las encuestas se puede concluir lo siguiente:

- Para la pregunta: ¿Te gusta el siguiente poema? Dada la evidencia muestral los resultados arrojaron que existía una diferencia en cantidad muy marcada entre las personas que sí les gustaban los poemas artificiales y a los que no (74.7 % SI - 25.3 % NO para la pregunta 1 y 61.4 % SI - 38.6 % NO para la pregunta 3). Se esperaba que el resultado en favor del desagrado fuera superior considerando las limitaciones del modelo GPT-2 usado (small) en comparación al completo (extra large). Siendo la versión pequeña menos poderosa en general que la más grande.

- Para la pregunta: ¿Qué poema te gusta más? Dada la evidencia muestral sobre la comparación entre el poema humano y el artificial, el poema artificial ganó el agrado de los encuestados en casi 2 tercios del total, la parte crucial de la comparación viene en la diferencia de los poemas con versos de arte mayor (pregunta 2) donde el poema artificial le gustó casi al 76 % de los encuestados, en cambio en la comparación de poemas con solo arte menor (pregunta 4), poco más del 54 % prefirió el humano. Es sorprendente el resultado considerando que los poemas humanos provenían de poetas profesionales y fueron de menor agrado que los de la máquina.

En general viendo como los poemas artificiales tuvieron una gran aceptación para el público encuestado a pesar de ser generados por un modelo que no es el mejor de su tipo muestra como las máquinas se van acercando a perfeccionar tareas que antes se veían lejanas como la generación de contenido creativo que pueda agradar a las personas, considerada antes una tarea exclusiva de los seres humanos. Y cada vez salen modelos más y más potentes que perfeccionan este tipo de tareas. Aún es temprano para decir que estos modelos serán los artistas del futuro, los avances son evidentes como en este caso de generación de poesía donde en el caso del español es solo un pequeño vistazo.

Bibliografía

[1] Joseph Weizenbaum. 1966. ELIZA—a computer program for the study of natural language communication between man and machine. Commun. ACM 9, 1 (Jan. 1966), 36–45.

[2] McCulloch, W.S., Pitts, W.H.: A Logical Calculus of the Ideas Immanent in nervous Activity. Bulletin of Mathematical Biophysics 5, 115–133 (1943)

[3] Hubel DH, Wiesel TN, Receptive fields of single neurones in the cat's striate cortex., J Physiol 148:574-591. ,(1959)

[4] G. Van Houdt, C. Mosquera and G. Nápoles, .ª review on the long short-term memory model", Artificial Intelligence Review, vol. 53, no. 8, pp. 5929-5955, 2020.

[5] K. Smagulova and A. P. James, .ª survey on LSTM memristive neural network architectures and applications", The European Physical Journal Special Topics, vol. 228, no. 10, pp. 2313-2324, 2019.

[6] A. Gatt, E. Krahmer, Survey of the State of the Art in Natural Language Generation: Core tasks, applications and evaluation, Journal of Artifcial Intelligence Research (JAIR), volume 61, pp. 65-170, 2018

[7] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. *Attention is all you need. In Advances in Neural Information Pro-cessing Systems*, pages 6000–6010.

[8] Jason Phang, Thibault Févry, and Samuel R Bowman. Sentence encoders on stilts: Supplementary training on intermediate labeled-data tasks. arXiv preprint arXiv:1811.01088, 2018.

[9] Cheolhyoung Lee, Kyunghyun Cho, and Wanmo Kang. Mixout: Effective regularization to finetune large-scale pretrained language models. In ICLR, 2020

[10] Anders Krogh and John A. Hertz. A simple weight decay can improve generalization. In NeurIPS, 1992.

[11] Jeremy Howard and Sebastian Ruder. Universal language model fine-tuning for text classification. In ACL, 2018.

[12] P. Izmailov, D. Podoprikhin, T. Garipov, D. Vetrov, and A. Wilson, Averaging Weights Leads to Wider Optima and Better Generalization (2019)

[13] Pablo Gervás, WASP: Evaluation of different strategies for the automatic generation of spanish verse., In Proceedings of AISB'00 Symposium on Creative & Cultural Aspects and Applications of AI & Cognitive Science, pages 93–100, 2000, Birmingham, UK

[14] H. G. Oliveira, F. A. Cardoso, F.C. Pereira, Tra-la-Lyrics: an approachto generate text based on rhythm. In Procs. of 4th International Joint Workshop on Computational Creativity, pages 47–55, 2007, London, UK. IJWCC 2007

[15] H. C. Oliveira A. O. Alves Poetry from concept maps – yet another adaptation of PoeTryMe's flexible architecture. In Proceedings of 7th International Conference on Computational Creativity, ICCC 2016, Paris, France.

[16] A. Astigarraga, J. María Martínez-Otzeta, I. Rodriguez Rodriguez, B. Sierra and E. Lazkano, "Poet's Little Helper: A methodology for computer-based poetry generation. A case study for the Basque language", Proceedings of the Workshop on Computational Creativity in Natural Language Generation (CC-NLG 2017), pp. 2–10, 2017.

[17] K. Wang, J. Tian, R. Gao and C. Yao, The machine poetry generator imitating Du Fu's styles, 2018 International Conference on Artificial Intelligence and Big Data (ICAIBD), Chengdu, China, 2018, pp. 261-265.

[18] B. Liu, J. Fu, M. Kato and M. Yoshikawa, "Beyond Narrative Description: Generating Poetry from Images by Multi-Adversarial Training", Proceedings of the 26th ACM international conference on Multimedia, 2018.

[19] M. Hämäläinen and K. Alnajjar, "Generating Modern Poetry Automatically in Finnish", Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), 2019

[20] Liu, Z., Fu, Z., Cao, J., de Melo, G., Tam, Y., Niu, C. and Zhou, J., 2019. Rhetorically Controlled Encoder-Decoder for Modern Chinese Poetry Generation. Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, [online], pp.1992–2001.

[21] Eun-Soon You, Soohwan Kang, and Su-Yeon O, Automated Korean Poetry Generation Using LSTM Autoencoder,1st International Workshop on Computational Humanities and Social Sciences (Computing4Human 2020), 2020.

[22] Y. Liu, D. Liu and J. Lv, "Deep Poetry: A Chinese Classical Poetry Generation System", Proceedings of the AAAI Conference on Artificial Intelligence, vol. 34, no. 09, pp. 13626-13627, 2020.

[23] N. Köbis and L. Mossink, .ªrtificial intelligence versus Maya Angelou: Experimental evidence that people cannot differentiate AI-generated from human-written poetry", Computers in Human Behavior, vol. 114, p. 106553, 2021.

[24] Pierre Guillou, Portuguese GPT-2 small: a Language Model for Portuguese text generation (and more NLP tasks...), 2020.

[25] M. Santillan and A. Azcarraga, "Poem Generation using Transformers and Doc2Vec Embeddings", 2020 International Joint Conference on Neural Networks (IJCNN), 2020.

[26] J. Lin, Y. Gao and R. Chang, Çhinese Story Generation with FastText Transformer Network", 2019 International Conference on Artificial Intelligence in Information and Communication (ICAIIC), 2019.

[27] Q. Wang, X. Wang, W. Liu and G. Chen, "Predicting the Chinese Poetry Prosodic Based on a Developed BERT Model", 2021 IEEE 2nd International Conference on Big Data, Artificial Intelligence and Internet of Things Engineering (ICBAIE), 2021.

[28] Khanna, C., 2021. Byte-Pair Encoding: Subword-based tokenization algorithm. [online] Medium. Available at: <https://towardsdatascience.com/byte-pair-encoding-subword-based-tokenization-algorithm-77828a70bee0> [Accessed 20 November 2021].

Apéndice A

Inteligencia Artificial

A.1. ¿Qué es la Inteligencia Artificial?

La Inteligencia Artificial es una rama de las ciencias de la computación que se encarga de investigar la forma de dotar a las máquinas de un comportamiento parecido al de los humanos. Sus objetivos principales son que las máquinas imiten el comportamiento humano, para que realicen sus tareas de manera autónoma y sin supervisión. Otro es el poder interactuar con el ser humano de una forma fluida y no mecánica. Además, que se puedan adaptar y aprender de su entorno y sus acciones previas.
La definición de inteligencia artificial es un concepto muy abstracto y un campo muy abierto que está en constante crecimiento debido a los avances tecnológicos actuales, ha logrado resultados que en épocas anteriores (1970'S y 1980'S) solo se soñaban. Se ha apoyado del incremento del poder de procesamiento y las grandes cantidades de datos a los que se puede tener acceso para generar nuevos modelos y algoritmos que aprovechen estos factores.
Estos algoritmos que usan grandes cantidades de datos para entrenarse y aprender son conocidos como algoritmos de Machine Learning.

A.1.1. Antecedentes

El crear seres inteligentes y autónomos siempre ha sido uno de los sueños del ser humano, como evidencia de esto se pueden ver registros de seres mecánicos inteligentes que deben de cumplir alguna función. Por ejemplo, en la mitología griega Talos era un autómata gigante hecho en bronce que se encargaba de resguardar la isla de Creta de cualquier enemigo. Otro ejemplo es el mito de Galatea, una estatua en marfil tallada por el rey de Chipre Pigmalion, esta obra al final tomaría vida como un ser inteligente. Más adelante en el tiempo las historias como Frankenstein (1818) hablan de la creación de un nuevo ser vivo y las repercusiones que este trajo a su creador.

En el caso de las computadoras, la inteligencia artificial es la encargada de realizar la búsqueda del ser inteligente, en este caso se apoya de distintas herramientas, entre ellas los algoritmos de Machine Learning y las redes neuronales como modelo más popular en la actualidad, empleando multitud de modelos que han surgido a lo largo de la historia de las computadoras.

En 1936 es concebida la Máquina de Turing . El dispositivo, aunque teórico, es la base de las actuales computadoras y su funcionamiento lógico. Era capaz de realizar cualquier cálculo matemático si se representaba como un algoritmo.

En 1943 Warren McCulloh y Walter Pitts liberaron el artículo *A Logical Calculus of the Ideas Immanent in nervous Activity* [2] donde proponían una neurona como entidad lógica con las capacidades operativas similares a una Máquina de Turing. En su propuesta sentaron las bases de lo que serían las redes neuronales. Proponiendo que una red neuronal podría representar un conjunto de instrucciones. Propusieron la estructura de las neuronas artificiales actuales basándose en la biológicas. Además de comenzar a dividir a las neuronas según la función que cumplieran en la red.

En 1949 surge la teoría Hebbiana, que desde un punto de vista de redes neuronales habla del efecto en las neuronas que interactúan cerca una de otra y como estas interacciones generan cambios metabólicos, en el caso de una neurona artificial esto significaba el cambio en los pesos de las neuronas participantes.

En 1950 fue el año en que Alan Turing propuso una prueba para determinar si una máquina podía tener un comportamiento inteligente parecido al ser humano o no. Nombrada como "Prueba de Turing", este método consiste en tener dos personas (A y B) y un ordenador (C), la persona A realizará preguntas a mediante un monitor y un dispositivo de entrada de texto (teclado), B y C darán sus respuestas sin que A conozca quién es la máquina. Si A determina que las respuestas son indistinguibles de las que daría un humano, entonces pasará a la siguiente prueba que es mantener una conversación entre la máquina y el ser humano por un lapso de tiempo determinado, si logra pasar la prueba se determinará si la máquina es inteligente o no.

En 1951 Marvin Minsky implementó la primera red neuronal conocida como SNARC (Calculadora de Refuerzo Analógico Neuronal Estocástico en español), máquina patrocinada por la fuerza aérea, el funcionamiento era similar a las redes actuales.

En 1956 durante la Conferencia de Dartmouth (oficialmente Dartmouth Summer Reserch Project on Artifcial Intelligence), convocada por John McCarthy para trabajar en conjeturas sobre la inteligencia en las máquinas y donde se reunieron 10 investigadores de alto grado deseosos de abordar un campo misterioso y nuevo para explorar. En esta conferencia se acuñó por primera vez el término "Inteligencia Artificial", naciendo formalmente el campo y consolidando así las ideas abordadas desde los inicios de la humanidad de crear entes autónomos y con pensamiento similar al de sus creadores, combinadas con los avances computacionales en la materia.

En la primera etapa, comprendida de 1956 a 1970, el objetivo de todos los investigadores se enfocó en desarrollar algoritmos complejos que resolvieran problemas específicos, por ejemplo, encontrar la ruta óptima, en este tiempo surgieron términos como "backtracking" para la búsqueda en laberintos. También es esta época surgieron las "reglas de producción" y los chatbots.

En 1958 Frank Rosenblatt crea la estructura del perceptrón, el cual es la unidad estándar en las redes neuronales actuales, usándolo por primera vez para el reconocimiento de patrones. Tiempo después Frank analizará la posibilidad de una red de múltiples capas.

En 1959 Wisel y Hubel publican "Receptive fields of single neurones in the cat's striate cortex" [3] y muestra como las neuronas se organizan en la corteza visual y así decodifican la información que le transmiten los ojos para convertirla en imágenes. Siendo clave tiempo después para el desarrollo de las redes convolucionales (usadas en el procesamiento de imágenes), las cuales intentan imitar el proceso de visión.

En el año 1969 Minsky y Papert publican el libro "Perceptrons", donde señala las limitaciones tecnológicas para usar el perceptrón, ademas muestra las deficiencias del modelo en la representación de la operación XOR, debido a que el perceptrón sólo realizaba operaciones lineales. El libro tuvo tal impacto en el campo que las investigaciones sobre redes neuronales que las organizaciones detuvieron todos los financiamientos a este tipo de investigaciones dejando varado el campo durante un tiempo.

En este periodo la inteligencia artificial tuvo problemas para desarrollarse, principalmente por el poco poder computacional de la época y los elevados costos de estas tecnologías, que restringen todos los avances. Las altas expectativas puestas en la IA combinadas con los pobres resultados obtenidos, la posicionaron como una opción poco atractiva para continuar invirtiendo recursos.

La segunda etapa que va del año 1970 al año 1980, se le conoce como el "Primer invierno" de la Inteligencia Artificial, debido a que se había perdido mucho del financiamiento ganado anteriormente y la credibilidad sobre el potencial de la IA había quedado en tela de juicio, disminuyendo el interés de las personas en el campo. Al depender tanto del poder computacional muchas investigaciones quedaron varadas por no poder acceder a el.

Durante esa época, la mayoría de los científicos fueron fuertemente criticados dentro de su ambiente académico, debido a las dificultades y problemas de sus investigaciones, pero no cedieron a estas presiones y continuaron avanzando a pesar de la situación adversa.

El esfuerzo fue recompensado con el nacimiento de los sistemas expertos, centrados en resolver tareas específicas simulando el juicio experto de un humano. También nació PROLOG (1972) considerado el primer lenguaje de programación lógico con su uso extendido aún en la actualidad.

Paul Werbos, investigador y pionero de las redes neuronales recurrentes, descubrió en 1975 *Backpropagation*, una técnica de aprendizaje clave en las redes neuronales actuales, esta técnica permite modificar de manera automática los parámetros de una red neuronal y mejorar su desempeño. Antes de esto la alternativa común era el uso de fuerza bruta, cambiar al azar los parámetros de la red, siendo difícil obtener los valores adecuados dejando el trabajo a la suerte.

La tercera etapa, comprendida entre 1980 y 1987 y conocida como "El Boom", con los avances tecnológicos en los ordenadores y la industria, se comienzan a popularizar los sistemas expertos para la automatización de procesos de producción. Se desarrollaron diversos sistemas conversacionales entre hombre y máquina.

Durante este periodo se incrementó la popularidad de ideas como el Conexionismo, que busca explicar los procesos complejos con la formación de redes de neuronas y como sus interconexiones se asemejan a la sinapsis realizada por las neuronas biológicas generando respuestas complejas.

La cuarta etapa de la inteligencia artificial que abarca de 1987 a 1993, fué caracterizada por la evolución de los equipos de cómputo desplazando a equipos especializados debido a su menor costo y mayor poder de cómputo. El presupuesto perdido a finales de los 1960'S comenzó a regresar a las investigaciones y surgió el desarrollo de algoritmos "inteligentes".

El mantenimiento de los sistemas expertos se volvió muy costoso dejándolos como opciones cada vez menos viables. Comienza a desarrollarse un enfoque en la robótica con la

visión de automatizar diversos procesos en fábricas productoras. Otra vez las altas expectativas puestas en la IA sobrepasan la realidad del proceso de desarrollo. El financiamiento volvió a caer y las investigaciones comenzaron a disminuir.

En la década de 1990 comenzaron a surgir modelos de redes neuronales más complejos en este caso las redes neuronales recurrentes las cuales son usadas para el procesamiento de texto. Además de identificar en 1991 uno de los grandes problemas de las redes neuronales hasta ese momento, *el desvanecimiento del gradiente*, el cual evita el entrenamiento adecuado de la red dejándola estancada.

La quinta etapa, de 1993 a 2011, se caracteriza por el gran desarrollo del poder de cómputo (Ley de Moore), la Inteligencia Artificial se comienza a dividir y especializar para resolver problemáticas específicas. El surgimiento de los agentes inteligentes vuelve a causar atención en la industria. Luego de la victoria del computador de IBM (*Deeper Blue*) sobre el campeón en ese momento de Ajedrez, Garry Kaspárov en 1997, la inteligencia artificial retomó popularidad y el desarrollo de algoritmos específicos para la industria se incrementó, siendo estos algoritmos la base y apoyo de muchos sistemas en la actualidad.

En 1997 nacen las redes Long Short Term Memory (LSTM) como sucesor de las redes recurrentes y llegaron para resolver el problema del descenso del gradiente, usando mecanismos que permiten actualizar la memoria de la red permitiendo emular memoria a largo plazo. En 1998 se produce la primera Red Convolucional, usada en una base de datos de dígitos escritos a mano (MNIST) dando buenos resultados.

En la etapa actual el mundo está completamente globalizado el campo de la IA ha proliferado debido al fácil acceso a la información, el aumento del poder de cómputo y el aumento de inversión en el campo por parte de las corporaciones e instituciones más poderosas del mundo.

Han surgido nuevos campos como Deep Learning dentro del ya conocido Machine Learning, campo de la IA enfocado en hacer que las máquinas aprendan de forma automática, desarrollando redes neuronales más complejas y precisas. Al costo de grandes cantidades de datos y procesamiento los resultados en diversos campos como literatura o traducciones se asemejan cada vez más a los logrados por humanos.

El desarrollo y crecimiento de las GPU (Graphics Processing Unit) inicialmente para procesar el cálculo de gráficos por computadora y su posterior uso en el Machine Learning ha permitido incrementar el número de cálculos realizado disminuyendo el tiempo invertido, dando lugar al sugimiento de distintos equipos de GPU, habiendo distintas gamas de potencia y precisión, al coste de pagar grandes sumas de dinero por la adquisición de las GPU's de mayor precisión usadas para las tareas más complejas.

En 2017 nace el término Transformer en el campo del ML mencionado por primera vez en el artículo *Attention is all you need* [7], este tipo de modelo de redes neuronales es más complejo que otros previos y utiliza mecanismos avanzados conocidos como mecanismos de auto atención para obtener resultados suficientemente buenos para ser indistinguibles de resultados humanos y con esto han dominado gran parte de las marcas dentro estado del arte actual.

En febrero del 2019 se libera GPT-2 (Generative Pre-trained Transformer 2) uno de los modelos Transformer más conocidos por su tamaño, potencia y continua aplicación para el procesamiento del lenguaje. Siendo el primero de los modelos en rebasar los mil

millones de parámetros, dando pauta a modelos cada vez más grandes y complejos.
Es creada una nueva clasificación para la Inteligencia Artificial, IA débil (capaz de resolver problemas específicos) y la IA fuerte (capaces de adaptarse a su entorno para realizar múltiples tareas).

A.1.2. Aplicaciones de la Inteligencia Artificial

Las aplicaciones de la inteligencia artificial se encuentran en todos los campos de la vida cotidiana en mayor o menor medida, desde estufas automáticas, casas inteligentes, sistemas automatizados, videojuegos, entre otros. Algunas de las aplicaciones en la actualidad son las siguientes.

- Dispositivos inteligentes: relojes digitales con múltiples funciones, electrodomésticos inteligentes (regular temperaturas de la nevera por ejemplo o lavadoras que ajustan los ciclos de lavado), luces inteligentes.
- Asistentes personales: Cortana de Microsoft por ejemplo o Siri de Apple
- Sistemas de seguridad: Sistemas de reconocimiento de huellas digitales, de retina de rostros, etc. Monitores de movimiento. Detectores de incendios.
- Lenguaje: generadores de informes, traductores, generadores de textos, análisis de intenciones, chatbots, etc.
- Videojuegos: inteligencia de los NPC, generación de mapas, renderizado automático de gráficos. Ambientes dinámicos.
- Visión: clasificación de imágenes, generación de imágenes, generación de texto mediante imágenes y viceversa.
- Sonido: Análisis de melodías, clasificación de melodías, generación de música, etc.
- Robótica: Robots inteligentes (drones de monitoreo o robots de rescate, por ejemplo), dispositivos mecánicos (como brazos robot para personas discapacitadas), robots industriales de línea de ensamblaje.
- Sistemas inteligentes: Automóviles con auto conducción, casas inteligentes, sistema de administración automáticos (Kubernetes con su sistema de manejo de carga), buscadores de Internet (Google), etc.
- Medicina: simulaciones inteligentes de enfermedades, análisis médico, diagnóstico de enfermedades.
- Química: análisis químico, simulación, clasificación.
- Generación de perfiles, generación de recomendaciones, clasificación de correo electrónico, generación de contenido.

A.1.3. Machine learning

El Machine Learning es un subcampo de la inteligencia artificial que permite a una computadora, mediante distintas técnicas y algoritmos, dotarla de la habilidad para aprender sin ser programada explícitamente. Su objetivo es diseñar sistemas autorregulados y para lograr esto se usan grandes cantidades de datos que serán empleadas para realizar predicciones.
Al ser parte del campo de la inteligencia artificial sus aplicaciones son las mismas, pero

se enfoca más en tareas que requieran el aprendizaje automático y donde se tenga gran cantidad de datos. Los algoritmos de machine learning hacen uso de conjuntos de datos relacionados al problema que deben resolver para aprender a predecir. A estos se les conoce como datos de entrenamiento. El entrenamiento en estos algoritmos es fundamental ya que, sin él, los algoritmos son inservibles. Además, el desempeño puede depender de la complejidad del problema, de los recursos disponibles o del enfoque adquirido para resolver el problema.

A.1.4. Tipos de algoritmos

Los algoritmos de Machine Learning los podemos dividir según el funcionamiento del algoritmo y si existe intervención de las personas.

- Aprendizaje supervisado: En este caso, se le proporciona al algoritmo un conjunto de datos previamente procesado para tener los datos ya clasificados (clases etiquetadas). Este tipo de algoritmos es más utilizado para la parte de clasificación. Como puede ser redes neuronales, máquinas de soporte vectorial, k-vecinos, etc.

- Aprendizaje no supervisado: Los algoritmos son más complejos, ya que deben de aprender de datos que no están ordenados como en el aprendizaje supervisado. Tampoco interviene el ser humano en su aprendizaje más allá de darle los datos. Debido a esto es frecuente que necesite más cantidad de datos para obtener resultados precisos. El algoritmo intentará encontrar la información para organizarla (autoorganización). Un ejemplo son los algoritmos de clustering.

- Aprendizaje por refuerzo: En este tipo de aprendizaje le proporcionan al algoritmo un incentivo (recompensa) cada vez que realiza una acción de manera correcta. En este tipo de aprendizaje el algoritmo está en monitoreo constante, mientras se busca la solución óptima para realizar un proceso. Se basa en la psicología conductista donde el agente aprende mediante prueba y error (recompensa y penalización).

Otra forma de clasificar es de acuerdo a su técnica de aprendizaje.

- Aprendizaje por lotes: El entrenamiento del algoritmo usa todos los datos disponibles y si se quiere usar nueva información para actualizar la red se debe de volver a entrenar desde cero. Es normal que el entrenamiento tome mucho tiempo (varias horas) debido a los grandes lotes de datos que manejan, además de requerir mucho poder computacional para realizar los cálculos.

- Aprendizaje incremental u online: Se caracteriza por el entrenamiento constante del algoritmo donde se busca que el modelo se ajuste al contexto del problema y se siguen proporcionando datos al proceso de entrenamiento de forma continua, pero al momento de agregar más datos el algoritmo no necesita iniciar desde cero el entrenamiento, en esta técnica el modelo previamente entrenado usa los nuevos datos para ajustar sus valores y mantener la precisión. Se usa cuando el conocimiento se necesita actualizar constantemente, ya sea por la naturaleza del problema o por los recursos disponibles para procesar los datos.

Un último tipo de clasificación es de acuerdo a la forma de categorizar los datos.

- Instancia: En esta clasificación el modelo predice las clases de nuevos ejemplos usando sus similitudes con los ejemplos de entrenamiento.

- Modelo: Se analizan los datos y sus patrones para luego seleccionar como modelar el problema que mejor se ajusten a los datos según el número de atributos que piensas elegir.

A.1.5. ¿Qué es Deep learning?

Dentro de los algoritmos de ML vistos anteriormente se encuentra un grupo especial que es conocido como algoritmos de Deep Learning (DL) o *Aprendizaje profundo*. El término *deep* o profundo se refiere al nivel de abstracción que pueden llegar a tener estos algoritmos a diferencia de otros, son capaces de aprender patrones y características complejas en los datos.
Por ejemplo, el rostro de una persona, un algoritmo de deep learning puede obtener patrones como la forma del rostro, la posición de los ojos o la nariz, características como el tipo de mirada, etc. Para realizar estas tareas se requieren algoritmos con estructuras complejas capaces de relacionar, filtrar y obtener patrones.
En general son redes neuronales, pero de muchas capas ocultas, se considera profundo desde 3 o 5 capas. Y así obtener la mayor cantidad de información y características de los datos. Debido al tamaño de estas redes, la porción de datos necesaria para su entrenamiento debe ser amplia y de calidad para evitar obtener información que podría afectar los resultados.
Las arquitecturas de redes neuronales se pueden enfocar en tareas específicas como son las redes convolucionales para el procesamiento de imágenes, las redes recurrentes para el procesamiento del lenguaje o autoencoders para aplicaciones como deepfake. Incluso se pueden combinar para crear arquitecturas de redes más complejas.

A.1.6. Diferencias Machine Learning y Deep Learning

Se debe señalar que los algoritmos de DL son un tipo de algoritmos de ML. Los algoritmos de DL son más complejos y más robustos, aunque la cantidad y calidad de la información puede afectar qué tan buenos resultados pueden alcanzar.
Además de necesitar mucho poder computacional para ejecutar los modelos más complejos de redes neuronales. Los dos enfoques comparten el mismo objetivo que es el aprendizaje automático de las máquinas en búsqueda de imitar el aprendizaje del ser humano. Otra diferencia es el uso exclusivo de redes neuronales como algoritmos bases de todos sus modelos por parte del Deep Learning, mientras que el Machine Learning puede hacer uso de estas redes además de una variedad muy grande de algoritmos de aprendizaje automático como Bayes, máquinas de soporte vectorial, k-vecinos, etc.

Apéndice B

Redes neuronales

B.1. ¿Qué es una red neuronal?

Son técnicas o mecanismos de aprendizaje que pretenden simular el aprendizaje de los seres biológicos. Una neurona biológica está compuesta de tres partes, axón, soma y dendritas. Las dendritas cubren el cuerpo de la neurona, parecen ramos pequeños que sobresalen del cuerpo de la célula, el soma es el cuerpo que contiene su núcleo y el axón al igual que las dendritas es una rama que sobresale de la neurona, pero esta puede ser muchas veces más larga que estos últimos, llega a medir 1 metro de largo. El axón se encarga de conectarse con otra neurona para recibir impulsos electro químicos, lo logra conectándose a las dendritas de otras células mediante su terminación.
La primera neurona genera un potencial de acción proveniente de impulsos externos, como sonido, dolor, temperatura, etc. Navega a través del axón hasta llegar al final de este, donde el potencial genera neurotransmisores, los neurotransmisores viajan mediante la conexión entre el axón de la neurona emisora y la neurona receptora. La conexión es conocida como sinapsis. Durante esta conexión la neurona receptora recibe la información de la neurona emisora en forma de estímulo químico, existen al menos 60 distintos tipos.
Una red neuronal artificial comparte similitudes con las redes biológicas, se puede ver a cada nodo (perceptrón) de la red como una neurona individual, la cual tiene entradas y salidas, la conexión y operaciones realizadas entre los nodos de la neurona artificial son una analogía al proceso de sinapsis que se genera en una neurona biológica.

B.2. Elementos de una red

B.2.1. Perceptrón

Unidad de procesamiento fundamental en las redes neuronales, inventada en 1958 por Frank Rosenblatt es un clasificador binario (sólo se pueden dos respuestas) está compuesta por:

- datos de entrada x: valores numéricos del problema.
- pesos w: valores que multiplican a cada dato.
- función de activación: función que se encarga de devolver una salida de acuerdo a los valores de entrada representa como se clasificaron los datos ya que devuelve salidas dentro de un rango determinado.
- Salida y: resultado que representan la predicción de acuerdo a los datos de entrada.

La Figura B.1 muestra que un perceptrón puede tener múltiples entradas pero solo una salida, además, para obtener esta salida todos los valores se suman (como se ve en la ecuación B.1 para n datos de entrada) y la salida es determinada de acuerdo a si el resultado sobrepasa o no cierto umbral dado por la función de activación.

$$\sum_{i=1}^{n} w_i x_i \tag{B.1}$$

Al valor resultante se le aplica una función de activación.

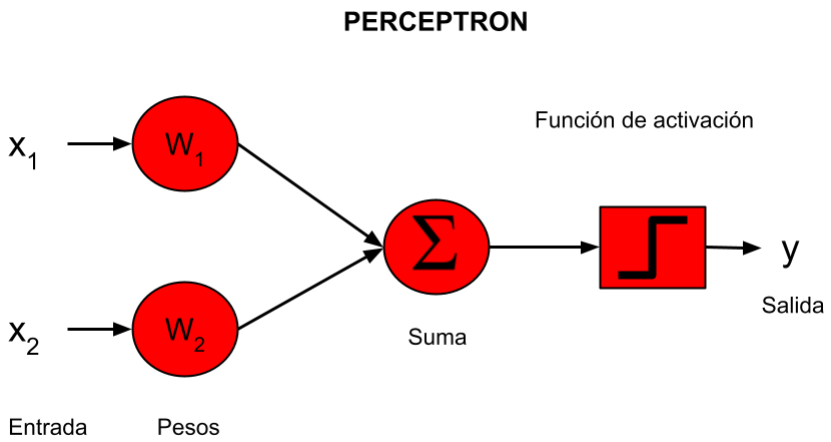

Figura B.1: Diagrama de un Perceptrón simple

B.2.2. Capas

Una capa se entiende como un grupo de neuronas o nodos que están al mismo nivel en la red neuronal. Por lo general se dividen en tres tipos

- Capa de entrada: Es la primera capa de la red la cual recibirá todos los datos de entrada.
- Capa de salida: Capa que tendrá a todas las neuronas que generan las salidas de la red.
- Capa oculta: Capas intermedias de la red pueden ser el número deseado de capas, mientras más capas ocultas tenga la red, más complejas se volverá. Una red con muchas capas ocultas se conoce como red profunda.

Todas estas capas están conectadas como se puede ver en la Figura B.2 y los datos se propagan a través de toda la red hasta llegar a los nodos destino que se encuentran en la capa de salida. Las redes de múltiples capas también son conocidas como perceptrón multicapa.

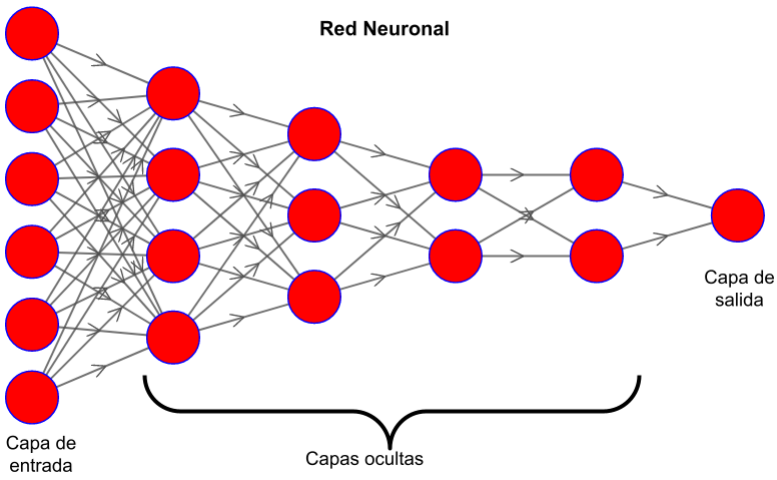

Figura B.2: Diagrama de una Red Neuronal

B.2.3. Pesos

Cada entrada en una red neuronal tiene un valor asociado a ella que se conecta con la siguiente neurona que representa la importancia de la entrada en la red. Esos valores son conocidos como pesos de la red. Durante el entrenamiento estos valores se ajustan de manera automática de acuerdo a los cálculos del error, mientras más valioso el peso, la entrada tendrá más impacto en el error. En la implementación normalmente se asignan valores aleatorios para iniciar.

B.2.4. Bias o sesgo

Bias o sesgo es un valor que se agrega a la suma ponderada en algunas redes neuronales permitiendo que se activen más veces algunas neuronas que otras. Normalmente cuando se necesita superar un umbral para asignar un valor, el sesgo puede facilitar que esto pase. Al hacer el ajuste del error también se tiene que ajustar el sesgo.
Representado en la suma ponderada, como en la ecuación B.2 para una neurona.

$$\sum_{i=1}^{n} w_i x_i + s \tag{B.2}$$

Donde

- x_i: datos i de entrada a la neurona.
- w_i: peso asociado al dato i de la neurona.
- s: sesgo de la neurona.

B.2.5. Función de Activación

Una función de activación se encarga de que la salida de una neurona esté entre un intervalo especifico. Su comportamiento es distinto dependiendo de qué tipo de función se aplique, por ejemplo se usa la función de tangente hiperbólica (Tanh) para decidir entre dos opciones, en cambio, una función sofmax devuelve una salida entre 0 y 1 que se puede interpretar como las probabilidades que ocurra cada opción del problema. En especifico se usa la función Tanh en las redes LSTM como una compuerta para decidir qué memoria mantener en la red y cual no.

B.2.6. Función de coste

Función que refleja la diferencia entre los valores reales y lo predicho por la red durante su entrenamiento de acuerdo a los datos de entrada, se le puede considerar el error de la red y se intenta minimizar durante el entrenamiento. Es descrito con frecuencia como el error cuadrático medio que estaría representado por la ecuación B.3.

$$C = \frac{1}{n}\sum_{i=1}^{n}[y_i - a_i]^2 \tag{B.3}$$

Donde
n: es el número total de ejemplos de entrenamiento
y_i: el valor correcto para el ejemplo i del conjunto de entrenamiento.
a_i: es el valor calculado para el ejemplo i en la salida por la red (predicción).

B.2.7. Backpropagation

Para saber como ajustar los pesos en una red neuronal durante el entrenamiento se debe obtener primero la salida de la red y calcular el error asociado con la función de coste. Luego se aplica un algoritmo conocido como backpropagation o "Propagación hacia atrás", el cual calculando las derivadas parciales del error asignará una parte del error a cada peso y sesgo de acuerdo a la contribución provocada por la entrada en el error final.

Descenso del gradiente

El descenso del gradiente es un algoritmo diseñado para encontrar el valor mínimo local en una función mediante el cálculo de la pendiente (primera derivada).
Para espacios de múltiples dimensiones se deben calcular todas las derivadas parciales necesarias, formando un vector gradiente que indicará la dirección donde se maximizará el valor, ahora como el objetivo es minimizar la función de error, se debe ir en dirección contraria a la del gradiente. Y se representa con la ecuación B.4

$$\theta_{i+1} = \theta_i - \alpha * grad_f(i) \tag{B.4}$$

Donde
θ_i es el punto (vector para múltiples dimensiones) de posición donde se encuentra la función en el momento i.
θ_{i+1} es el siguiente punto.
α es el factor con el que se va a vamos a mover el gradiente, es análogo al largo de los pasos de una persona al caminar.
$grad_f(i)$ el vector gradiente en el punto i
Si se dibuja el relieve de una función, el descenso del gradiente buscará el punto más bajo más cercano de acuerdo a su gradiente.

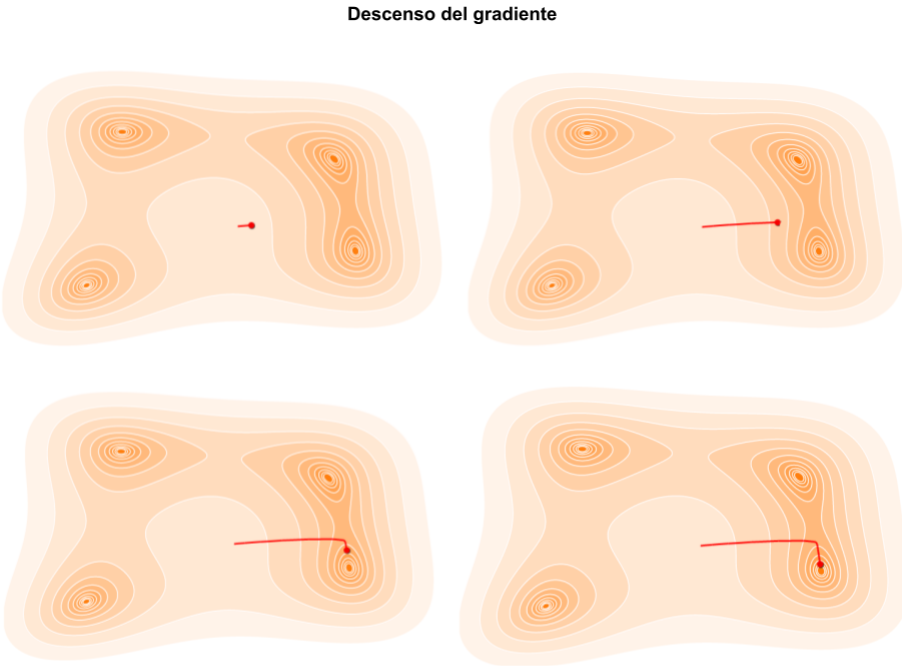

Figura B.3: Visualización del descenso del gradiente

En la Figura B.3 se puede ver como el punto se mueve desde su posición inicial hacia el mínimo más cercano, aunque no se puede visualizar el valor de α, este debe de ser lo suficientemente grande para evitar tardar mucho tiempo y debe de ser suficientemente pequeño para no saltarse el óptimo.

Regla de la cadena para Backpropagation

La regla de la cadena se usa para facilitar el cálculo de las derivadas parciales durante la propagación hacia atrás. La regla para el caso de varias variables consiste en lo siguiente
Sea **w** = f(x,y) una función diferenciable de **x** e **y**,
además x = g(t), y = h(t) funciones diferenciables de **t**.
w entonces es función de t y se puede ver como w = f(x(t), y(t))

$$\frac{\partial w}{\partial t} = \frac{\partial w}{\partial x}\frac{\partial x}{\partial t} + \frac{\partial w}{\partial y}\frac{\partial y}{\partial t} \tag{B.5}$$

Procedimiento de Backpropagation para una neurona cualquiera en una capa cualquiera

Paso 1: Obtener la salida de la red y calcular su error con la función de coste. Aunque existen diversas funciones para calcular el error, en este caso se empleara el error cuadrático medio, mismo de la ecuación B.3.
Paso 2: Calcular calcular el error asociado al peso w de la neurona i en la capa n.

$$\frac{\partial C}{\partial w_i^n} \tag{B.6}$$

Donde

- C: Error total de la red.
- w_i^n: Peso de la neurona i en la capa n.
- s_n^i: sesgo de la neurona i en la capa n.

Para obtener el error de una neurona en la capa n, se debe obtener la derivada parcial del error con respecto a la neurona en esa capa dado por la ecuación B.7.

$$\frac{\partial C}{\partial w_i^n} = \frac{\partial C}{\partial a^n} * \frac{\partial a^n}{\partial z^n} * \frac{\partial z^n}{\partial w_i^n} \tag{B.7}$$

Donde

- C: Error total de la red dado por la función de coste.
- w_i^n pesos de la neurona i en la capa n de la red.
- a^n función de coste de la neurona en la capa n.
- z^n suma de los pesos de la neurona en la capa previa.

la derivada parcial del error con respecto de la función de activación en el caso de la última capa se puede ver como la derivada parcial en la función del coste, en este caso se puede ver el resultado en la ecuación B.8.

$$\frac{\partial C}{\partial a_i^n} = \frac{\partial}{\partial a_i^n} \frac{1}{2} \sum_{i=1}^{T} [y_i - a_i^n]^2 = (y_i - a_i^n) \tag{B.8}$$

Donde

- y_i es el valor correcto para los datos de entrada de el ejemplo i.
- a_i^n es la predicción en la capa n para los datos del ejemplo i.

Para cualquier neurona en la red, la derivada de la función de activación en la capa n con respecto a la suma de los pesos de la capa n $\frac{\partial a^n}{\partial z^n}$, depende de la función de activación que se use en la capa específica.
Para la derivada de la suma de los pesos en la capa n con respecto a los pesos de la capa n $\frac{\partial z^n}{\partial w^n}$ se tiene la siguiente ecuación B.9.

$$\frac{\partial z^n}{\partial w_i^n} = \frac{\partial}{\partial w_i^n} \sum_{i=1}^{n} w_i a_i^{n-1} + s = a_i^{n-1} \tag{B.9}$$

Donde

- w_i^n es la neurona i en la capa n.
- a_i^{n-1} es el valor del dato de salida de la capa anterior o $n-1$ y que recibe la neurona i en la capa n.

Ahora el resultado se le asigna al peso como se hace en descenso del gradiente

$$w^n = w^n - \alpha \frac{\partial C^n}{\partial w^n} \tag{B.10}$$

Y se repite para todos los pesos y sesgos en la red hasta que se ajusten.

Desvanecimiento del gradiente y explosión del gradiente

Cuando se usa una red con muchas capas y se quieren ajustar sus pesos, existen situaciones en las que el valor de las derivadas parciales que llega a las primeras capas es tan pequeño que no permite cambios significativos , y la red queda estancada sin poder mejorar, por eso se dice que se desvanece el gradiente. el caso contrario es cuando el valor del gradiente comienza a aumentar conforme recorre las capas hacia atrás, llegando a modificar tanto los pesos que la red comienza a fallar y es imposible entrenarla.

B.3. Redes Neuronales Convolucionales

Las redes neuronales convolucionales suelen usarse para trabajar imágenes, por ejemplo para clasificarlas u obtener algunas características de estas. Lo que recibe es la imagen mapeada a sus componentes RGB en una matriz, si la imagen es en banco y negro solo se necesitará una matriz por imagen, si es a color, se deben de usar 3.
Los principales elementos de una red convolucional son:

- Filtro: También conocido como kernel, es una matriz pequeña que opera punto a punto con la matriz de la imagen, permitiendo obtener las características más importantes de la imagen. Los filtros pueden ser iniciados de forma aleatoria o con formas específicas para que obtengan características deseadas.
- Convolución: Operación punto a punto entre un filtro y una imagen (se realiza por áreas del mismo tamaño que el filtro). El filtro se comienza a aplicar desde la esquina superior izquierda de la matriz de la imagen. El resultado de la convolucion es otra matriz conocida como *mapa de características* y representa las características obtenidas de la imagen por el filtro.
- Padding: Consiste en agregar elementos (valores numéricos) a los bordes de la matriz de la imagen para conservar las medidas en los mapas de características cuando se aplique el filtro. Cuando se agregan ceros en los bordes, se conoce como *zero padding*.
- Stride: Es la longitud de salto en la convolución. Luego de aplicar el filtro en un área de la matriz, el filtro debe moverse a la derecha o hacia abajo cierta cantidad de píxeles, esa distancia la determina por el Stride.
- Max Pooling: Operación que se puede aplicar la la matriz resultante de la convolución para obtener la información más importante de esta, consiste en tomar los valores más grandes de cierta área con una medida establecida y así reducir las dimensiones sin perder la información más importante.
- Stacking: Es el apilamiento de varios mapas de características para obtener estructuras más complejas.

Un pequeño ejemplo de una red convolucional donde se aplican algunos de estos mecanismos es la siguiente Figura B.4.

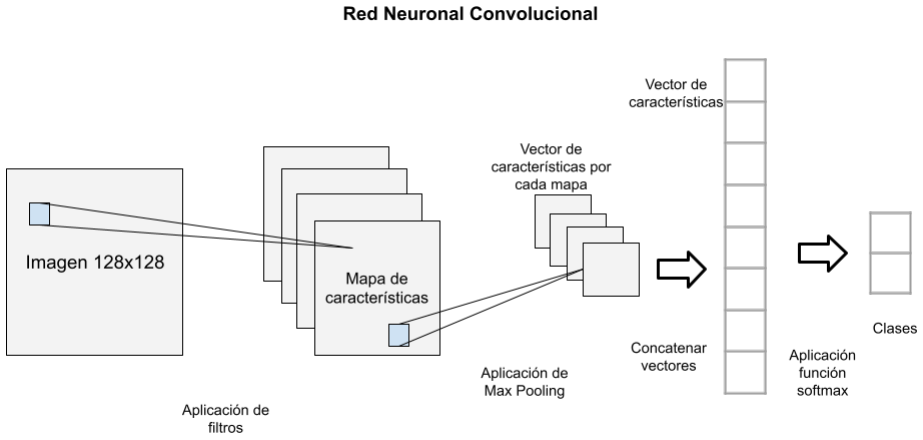

Figura B.4: Ejemplo de Red neuronal Convolucional

B.4. Redes Neuronales Recurrentes

Las redes neuronales recurrentes tienen capacidades especiales que la separan de las redes básicas. Primero, es capaz de trabajar con secuencias de datos relacionados a diferencia de las básicas que solo trabajan dato a dato sin considerar su relación. Segundo, para que la característica anterior sea efectiva deben de tener una memoria capaz de recordar la relación entre elementos dentro de la secuencia. Por eso se dice que estas redes tienen memoria a corto plazo.
Estas redes se utilizan comúnmente para el procesamiento de texto debido a que el texto es una secuencia de palabras o caracteres con una relación de orden entre ellos. Cada vez que entra a la red un nuevo elemento perteneciente a la secuencia, se conoce como *instante de tiempo*.

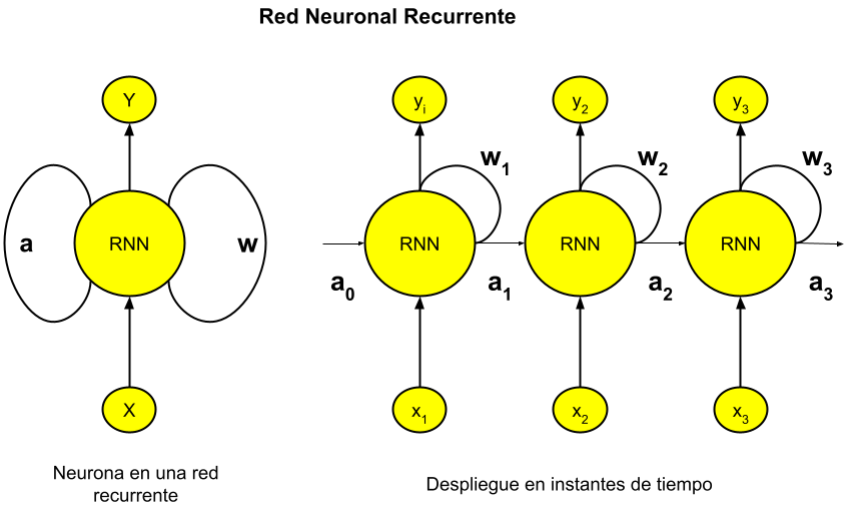

Figura B.5: Ejemplo de Red neuronal Recurrente

En la Figura B.5 se ve la representación habitual de las redes recurrentes, en lugar de neuronas las unidades que utiliza se conocen como *estados ocultos* por un lado se ve la red con sus respectivas entradas y salidas aunque mostradas de forma recurrente y por otro se ve su forma desplegada a cada instante de tiempo.
La neurona recibe como entrada el estado oculto de la etapa previa a_{t-1} y la entrada es el elemento correspondiente de la secuencia x_t.
Para calcular la función de activación del estado oculto actual se usa la fórmula siguiente

$$a_t = f(w_a a_{t-1} + w_x x_t + s_a) \tag{B.11}$$

Y para calcular la salida y_t

$$y_t = g(w_y a_t + b_y) \tag{B.12}$$

La memoria a corto plazo se encuentra en las operaciones necesarias para obtener a_t y y_t ya que ambas dependen del estado oculto anterior a_{t-1}. Los valores w_a,w_x,w_y así como s_a y s_y se ajustarán parecido a una red neuronal que usa backpropagation.

B.5. Redes Long Short-Term Memory

Son redes diseñadas específicamente para evitar el problema del desvanecimiento del gradiente. Lo logran mediante una estructura compleja de **compuertas** que permiten el

control de la información que se aprende a cada paso (instante de tiempo) conservando lo mas relevante y desechando lo demás.
La parte clave en el modelo LSTM es la capacidad de modificar la información que transporta la celda. El medio de transporte de la información se conoce como **cell state**, y contiene los datos que serán pasados de instante en instante. El cell state participa en todo el proceso de memoria de la celda.
En primer lugar la celda LSTM recibe como entradas:

- Cell state del estado anterior (c_{t-1}): el cual lleva la información.
- Estado oculto anterior (a_{t-1}): Cumple el mismo objetivo que una celda RNN pero adaptado a una LSTM.
- Entrada (x_t).

Las distintas funciones de activación en la celda tienen cada una un propósito específico:

- Función Tanh(tanh): Transforma la entrada en valores entre -1 y 1, sirve para regular los valores en la red y no se disparen.
- Función Sig(σ): Transforma la entrada en valores entre 0 y 1. Actúa como una válvula, cuando el valor en el vector es cercano a 0 el valor se olvidará o no se tomará en cuenta, en el caso contrario cuando es cercano a 1 entonces el valor se mantendrá o actualizará. Todo depende de la compuerta en la que se encuentre

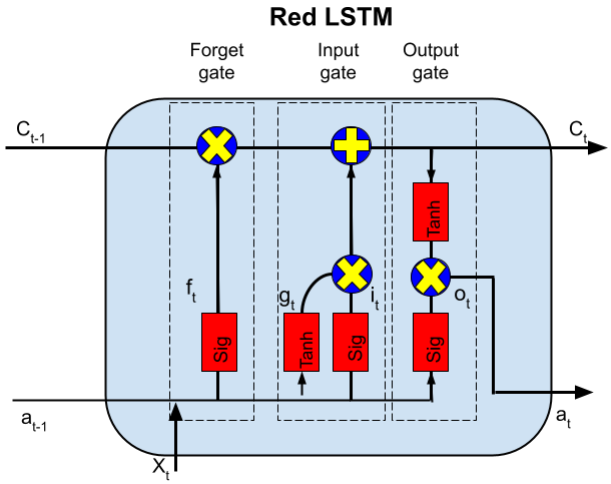

Figura B.6: Modelo de compuertas en una Red LSTM

Una celda LSTM se divide en tres compuertas diferentes [4]. Una se encarga de decidir qué información olvidar y cual mantener sobre los datos de entrada, otra sirve para añadir información nueva y la última sirve para generar las salidas de la red durante ese instante de tiempo.

Forget Gate

La compuerta Forget gate controla cuanta memoria se mantiene del estado pasado. Sus entradas son el estado oculto anterior a_{t-1} y el elemento nuevo de la secuencia x_t.
Se concatenan los dos vectores y luego se aplica una función sigmoidea, que de acuerdo a la entrada asignará los valores en el nuevo vector entre 0 y 1.
Los valores que deben de ser olvidados se transformaran en 0 y los que no en 1.
Luego el vector se multiplica punto a punto por el vector del *cell state* y así borra la

memoria innecesaria en la red.
Note que es solo para borrar los datos del estado anterior. Aún no se han ingresado nuevos datos. En la Figura B.6 el proceso está representado por f_t de la ecuación B.13

$$f_t \ = \ \sigma(w_f x_t \ + \ u_f a_{t-1} \ + \ s_f) \tag{B.13}$$

donde

- w_f: peso asociado al elemento x_t de la red en para el instante f.
- x_t: elemento t de la secuencia de entrada.
- u_f: peso asociado a la función de activación a_{t-1}.
- a_{t-1}: función de activación del instante $t-1$.
- s_t: sesgo de la red en el instante t.

Input Gate

La compuerta Input gate que sirve para añadir nueva información que será guardada en la memoria de la cell state.
Primero las funciones g_t y i_t reciben cada uno una copia de los datos a_{t-1} y x_t. La función sigmoidea en i_t, representada en la ecuación B.14, realiza el mismo proceso que la compuerta forget gate para separar la información que si se quiere mantener, pero ahora del estado anterior y de los nuevos datos y luego genera un vector nuevo.

$$i_t \ = \ \sigma(w_i x_t \ + \ u_i a_{i-1} \ + \ s_i) \tag{B.14}$$

Luego g_t genera un vector entre -1 y 1 de la combinación de las dos entradas x_t y a_{i-1}, para mantener una proporción en los valores que se van a pasar con respecto a los del estado oculto, como se ve en la ecuación B.15.

$$g_t \ = \ \tanh(w_g x_t \ + \ u_g a_{t-1} \ + \ s_g) \tag{B.15}$$

Se multiplican los dos vectores g_t e i_t para formar un vector completamente nuevo con los valores que actualiza, el cell state, y así se añade nueva información a la red.

Output Gate

Output gate es una compuerta que permite calcular el nuevo estado oculto a_t de acuerdo al cell state actualizado las dos entradas. Primero se pasan las dos entradas x_t y a_{t-1} por una función sigmoidea para obtener los valores que si pasaran al siguiente estado oculto a_t. Se aplica al vector de cell state, Ecuación B.17, una función tanh y así no perder las proporciones.
Ahora se multiplicarán los dos vectores salientes de cada función en o_t, Ecuación B.18, para transformarse en el nuevo estado oculto a_t que será una de las salidas junto al cell state c_t.

$$o_t \ = \ \sigma(w_o x_t \ + \ u_o h_{t-1} \ + \ s_o) \tag{B.16}$$

Este proceso se hace con cada elemento en la secuencia evitando el cargar con información innecesaria durante grandes periodos de tiempo. Además de mantener los valores ajustados a rangos establecidos en todo momento.

$$C_t \ = \ g_t \bigodot i_t \ + \ f_t \bigodot C_{t-1} \tag{B.17}$$

y

$$a_t \ = \ o_t \bigodot \tanh(C_t) \tag{B.18}$$

B.6. Redes Gated Recurrent Unit

Las redes GRU son una variante o simplificación [5] de una red LSTM. Primero, se tienen 2 compuertas en lugar de 3 combinando las compuertas Forget e Input en una nueva llamada Update (z_t) , la segunda compuerta llamada Reset gate (r_t) actúa para indicar cuanta información se debe olvidar.
Segundo, desapareció el cell state y solo tenemos como entrada x_t y a_{t-1}. Los estados C_{t-1} y a_{t-1} se convierten en a_{t-1} para GRU (Figura B.7). En el proceso se calcula un estado oculto parcial $h*_t$ que luego será combinado para obtener el nuevo a_t.
Solo tenemos una salida a_t. Las GRU pierden precisión por ganar velocidad en la ejecución debido al recorte de cálculos en relación a las LSTM.
Al igual que en en el modelo anterior, las compuertas se encargan de determinar la memoria a guardar y a borrar. Además la actualización de a_{t-1} hacia a_t es parecida a la del cell state de LSTM.

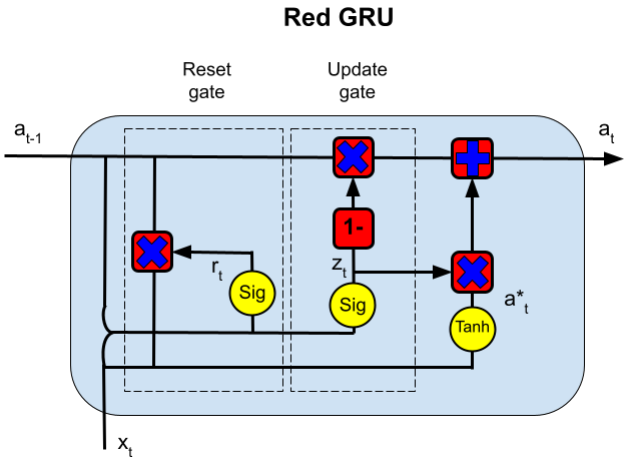

Figura B.7: Modelo de compuertas en una Red GRU

Reset Gate

La compuerta Reset gate sirve para determinar qué parte del estado anterior irá en el estado candidato $h*_t$.
Recibe como entrada el estado oculto anterior a_{t-1} y el elemento de la secuencia correspondiente x_t.

$$r_t = \sigma(w_r x_t + u_r a_{t-1}) \tag{B.19}$$

Update Gate

La compuerta Update gate tiene la responsabilidad de borrar la memoria e ingresar nueva a la celda, Se reciben dos entradas, a_{t-1} y x_t.
Se aplica una función sigmoidea y lo obtenido decidirá si el contenido será recordado u olvidado.
Si $z_t = 0$ entonces se borrará el contenido en a_{t-1} pero cuando se haga el cálculo de a_t se añadirán los valores nuevos. Si sucede que $z_t = 1$, entonces el contenido del estado anterior se mantiene intacto.

$$z_t = \sigma(w_z x_t + u_z a_{t-1} + s_z) \tag{B.20}$$

Ahora para calcular $a‘_t$ se utiliza el vector obtenido en r_t además de las dos entradas a_{t-1} y x_t

$$a‘_t = \tanh(w_{a‘} x_t + w_{a‘}(r_t \bigotimes a_{t-1}) + s_{a‘}) \tag{B.21}$$

Además la actualización final del estado se hace con

$$a_t \;=\; z_t \bigotimes a_{t-1} \;+\; (1 \;-\; z_t) \bigotimes a`_t \tag{B.22}$$

Apéndice C

Transformers

C.1. Transformers

Los Transformers son modelos de Machine Learning introducidos en 2017 [7] y enfocados en solucionar problemas de Natural Language Processing (NLP), ya sea para clasificación como análisis de reseñas o para modelado del lenguaje como generación de texto para tareas específicas.
Antes de su llegada la gran mayoría de los modelos de NLP eran variantes de las Redes Neuronales Recurrentes, LSTM, GRU, BRNN, etc. Las cuales están especializadas en trabajar con secuencias de datos. A pesar de esto, las redes de tipo recurrente sufren dos grandes problemas.
1. La imposibilidad de paralelizar el proceso secuencial: Durante el entrenamiento las redes recurrentes y sus variaciones procesan ordenadamente cada elemento de una secuencia, haciendo imposible paralelizar este proceso con cada secuencia individualmente. Siendo el entrenamiento un proceso de larga duración y costoso en términos de poder de cómputo.
2. La segunda es la dificultad de obtener el contexto total de cada elemento en la sentencia cuando ésta es demasiado grande. Por eso se suele trabajar con secuencias de tamaño moderado (10, 100, 1000 elementos). Otro problema que nace con esto es la dificultad de representar los distintos significados de una sola palabra a lo largo de la misma sentencia.

> Me lastimé la **planta** del pie por pasar descalzo sobre una **planta** con espinas y ahora no puedo subir las escaleras hacia mi habitación en la segunda **planta**.

Por ejemplo, el recuadro anterior resalta la palabras *Planta*, la cual puede aparecer en la frase múltiples veces, teniendo distintos significados, de acuerdo al uso que se de dentro de la frase. El primero se refiere a la parte baja del pie. La segunda se refiere a un ser vivo que subsiste mediante la fotosíntesis y la tercera se entiende como cada uno de los niveles en un edificio.
Los Transformers dan solución a ambos problemas al implementar nuevas estrategias y arquitecturas capaces de obtener el contexto completo y siendo paralelizables al trabajar con las secuencias completas al mismo tiempo. Debido a esta capacidad, los Transformers se entrenan con grandes cantidades de datos que con otros modelos sería imposible intentar debido a lo costoso que es y así frecuentemente logran alcanzar resultados al nivel de los reportados en el estado del arte.

C.1.1. Arquitectura

Un modelo Transformer tradicional se puede ver como varios bloques distintos interconectados realizando pequeñas tareas con el objetivo de generar una secuencia de salida en base a la secuencia de entrada recibida, como se puede ver en la Figura C.1.
Los elementos principales de un Transformer son:

- Input Embedding y Positional Encoding: Estas dos capas se encargan de transformar el texto de entrada en algo que pueda procesar el modelo.
- Encoder: Recibe los datos, los procesa y genera una representación vectorial de la relación entre cada elemento de la secuencia de entrada y la pasa al Decoder.
- Decoder: Recibe la secuencia objetivo y con ayuda de la salida del Encoder crea una representación de los datos de entrada para generar predicciones sobre el elemento siguiente en la secuencia objetivo.
- Módulos de Self-Attention: Bloques dentro del Encoder y el Decoder que permiten la captura completa del contexto para cada elemento en el vocabulario.

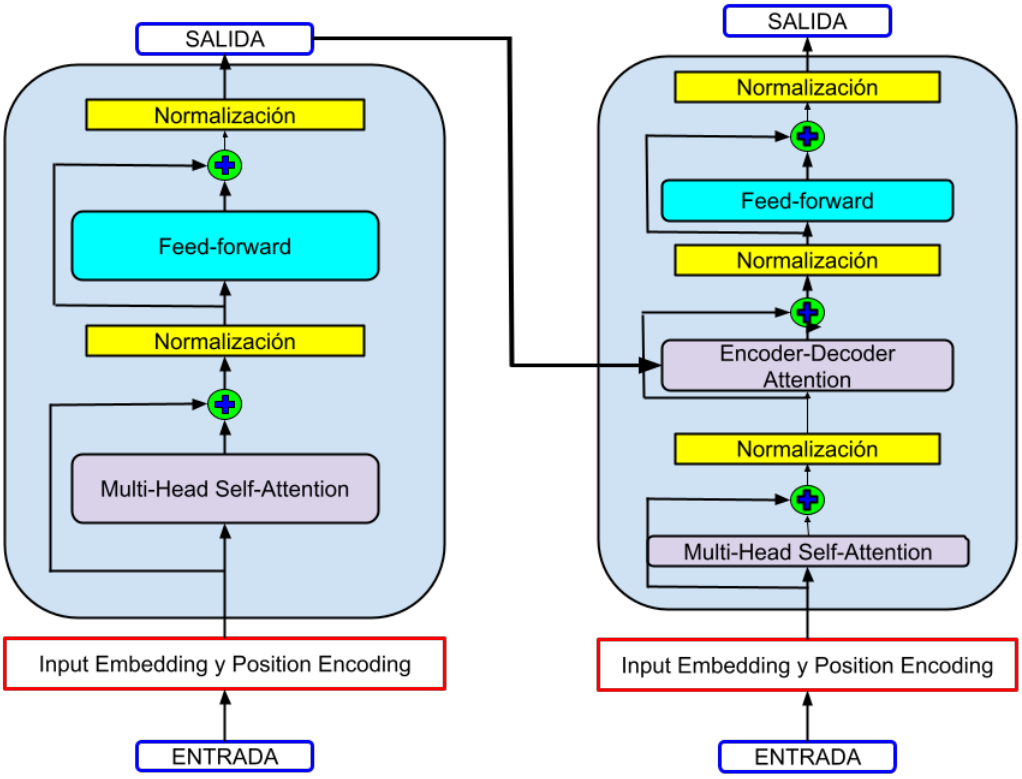

Figura C.1: Conexión amplificada entre en Encoder y un Decoder

C.1.2. Input Embedding y Positional Encoding

El modelo recibe como entrada Texto pre-procesado, es decir, a cada elemento, en el caso de los Transformers sub palabras, se asigna un índice que será su representación numérica en el vocabulario.
La representación es tomada como entrada por la capa **Input embedding**, la cual internamente genera una representación vectorial, al igual que lo haría un *word embedding*, y la llena con números aleatorios. El tamaño original usado para cada vector se estableció en 512 características. Se puede ver como una matriz donde estarán todos los elementos

del vocabulario, como se ve en la Figura C.2.
Al trabajar con una secuencia completa de datos, distinto a como sucede con las redes

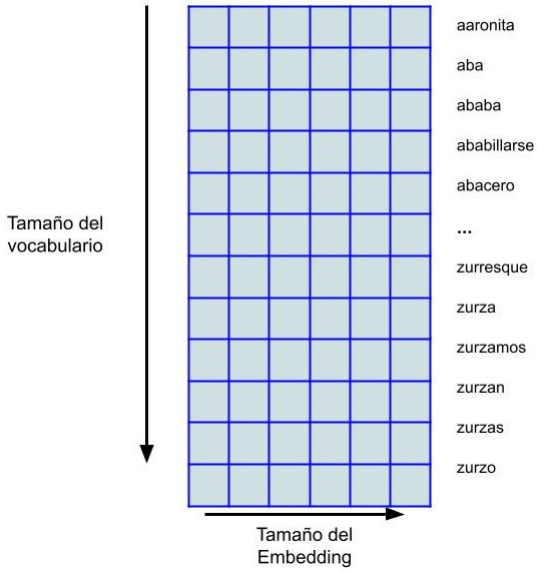

Figura C.2: Vista matricial del Embedding

recurrentes que trabajan de forma secuencial, es necesario saber la posición de cada elemento.
Para guardar la información del orden de cada elemento se usa un encoding posicional que es una estructura vectorial de las mismas dimensiones del input embedding, ambos se suman para generar vectores que representen cada secuencia y su orden al mismo tiempo obteniendo una representación única para cada elemento en la secuencia.
Como se puede ver en la Figura C.3, no toma en cuenta el tamaño del vocabulario, sino, el tamaño de la secuencia que está leyendo, representando así la posición para sus elementos.
Para representar de forma única cada posición se hace uso de las ecuaciones C.1 y C.2

$$PE(pos, 2i) = \sin(\frac{pos}{10000^{\frac{2i}{d_{model}}}}) \tag{C.1}$$

y

$$PE(pos, 2i+1) = \cos(\frac{pos}{10000^{\frac{2i}{d_{model}}}}) \tag{C.2}$$

Donde

- pos: posición del elemento en la secuencia.
- i: el índice que representa cada casilla en el vector embedding para cada elemento de la secuencia (va de 0 al tamaño del vector embedding -1).
- d_{model}: tamaño del vector embedding para el modelo.

Ahora el nuevo embedding se puede entender como un Embedding Posicional (EP) que es necesario para el siguiente elemento.

C.1.3. Mecanismo Self-Attention

¿Qué se entiende con Atención? Los mecanismos de atención permiten analizar la cercanía (en relación al contexto) de cada elemento de la secuencia de entrada entre sí. En el caso del mecanismo Self-Attention, su objetivo es enfocarse en un elemento en específico

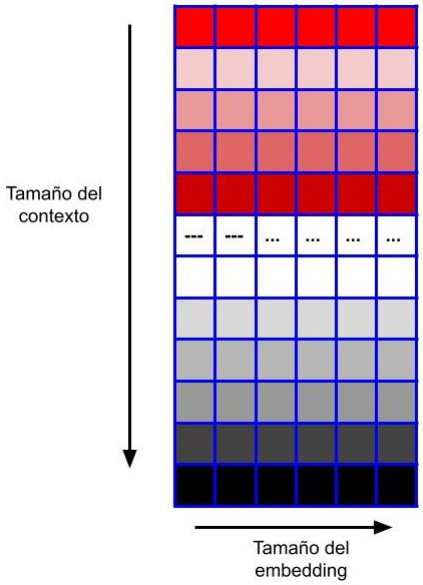

Figura C.3: Vista del Encoding Posicional

y buscar los elementos más relacionados. como esto puede cambiar entre sentencias para el mismo elemento. Self-Attention captura el contexto total, conocido como Full-context, para cualquier sentencia. Siendo en este punto superior a las redes recurrentes.

La forma de trabajo de este mecanismo es singular, ya que emplea con tres copias del

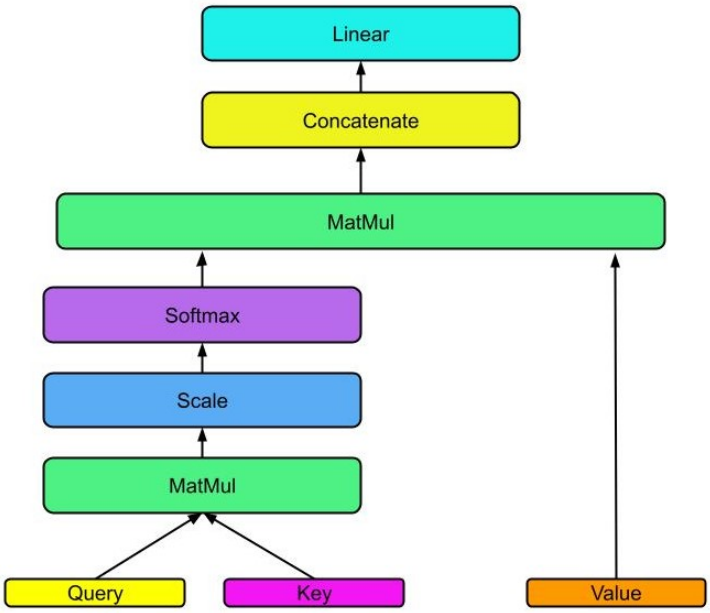

Figura C.4: Arquitectura de un Módulo de Atención

mismo Embedding Posicional. Cada copia es multiplicada por una matriz de valores que al inicio son aleatorios y se irán ajustando durante el entrenamiento y serán lo equivalente a los filtro de una red convolucional, dando como resultado tres nuevos vectores: Query, Key, Value. Los nuevos vectores tienen la misma dimensión entre sí, pero son distintos al Embedding Posicional original.

De acuerdo a la Figura C.4 el camino a seguir para un mecanismo de Self-Attention es:

- Se realiza el producto punto en los vectores Query y Value, el modelo está diseñado

para que las dimensiones de los vectores concuerden. En la capa **MatMul**, se fija un elemento y se realiza el producto punto contra todos los demás elementos, Figura C.5.

- Se escalan los valores en la matriz dividiendo por la raíz cuadrada de la dimensión de los vectores Query y Key.
- Se aplica una función Softmax a cada elemento en la matriz.
- El vector resultante, conocido como filtro de atención, se multiplica por el vector Values que no se ha tocado hasta este punto.
- Esta será la salida en un módulo de atención.

Pero no se debe olvidar que un Transformer implementa lo que se conoce como **Multihead-Attention**. Esto es que aplica diferentes filtros a diferentes módulos de Self-Attention. Todo al mismo tiempo y cada uno con sus propias copias. En el modelo original se usaron 8 cabezas, pero pueden ser más.
Al terminar todos los módulos sus salidas se concatenan y multiplican por una matriz, que también se ajusta durante el entrenamiento, para dar como resultado un nuevo vector que será enviado a la capa feed-forward.

	Ejemplo	de	como	se	veria	el	cálculo
Ejemplo	21	23	12	17	34	29	58
De	10	43	21	28	32	23	74
como	23	54	22	17	14	45	64
se	22	12	33	11	50	12	53
veria	31	11	56	26	70	29	37
el	19	12	12	27	54	26	23
cálculo	56	23	14	28	43	35	19

Figura C.5: Producto punto con cada palabra

C.1.4. Encoder

El bloque Encoder está formado de 6 o más Encoders conectados. Cada Encoder tiene internamente una capa de tipo Self-Attention además de una red Feed-forward. Cada una tiene una subcapa de normalización y conexiones residuales entre ellas, como se muestra en la Figura C.6, así se evita perder información importante de pasos anteriores, preservando la memoria y evitando problemas como desvanecimiento del gradiente. La función de la subcapa de normalización es recibir la entrada y la salida de la capa anterior, combina ambas y las normaliza. Permitiendo una mejor convergencia y estabilidad al modelo.
El primer Encoder en el bloque es el que recibe la secuencia ya procesada, ejecuta su funcionamiento interno y devuelve una salida al siguiente Encoder, el resultado de cada Encoder pasa al siguiente Encoder hasta terminar el bloque. El resultado final es enviada al Decoder.

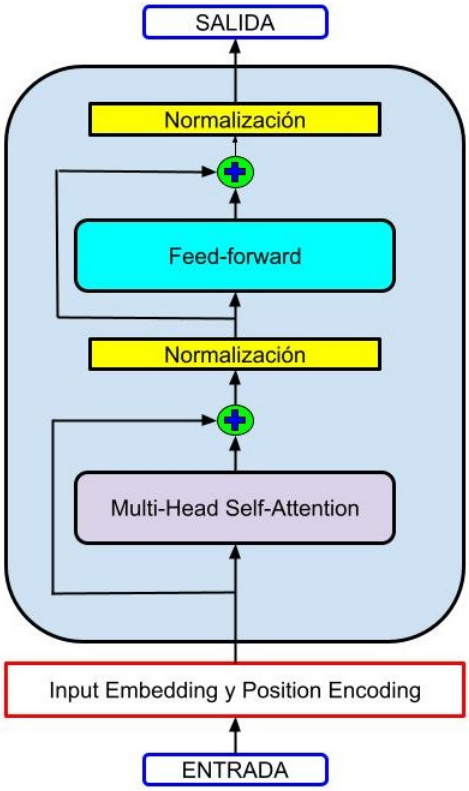

Figura C.6: Contenido interno del primer Encoder en el bloque

C.1.5. Decoder

El bloque Decoder se forma de 6 o más Decoders al igual que el bloque Encoder. El primero recibe una secuencia objetivo que es preprocesada usando el embedding posicional. Siendo parecido al Encoder, el Decoder tiene también los módulos de Self-Attention y Feed-forward. Las conexiones residuales y las subcapas para normalizar también están después de cada módulo.
Además de estos módulos agrega un módulo extra, como se ve en la Figura C.7, el nuevo módulo ayuda al modelo a recuperar las partes importantes en la sentencia.
El módulo extra recibe la salida del bloque Encoder y la salida del módulo Self-Attention y los combina, además de funcionar como un módulo extra de Self-Attention. Permitiendo así un mejor aprendizaje de las relaciones entre los elementos de la secuencia. Recibe 3 entradas, el vector Value del vector generado por el módulo Self-Attention anterior y los otros dos vectores, Query y Key se generan con la salida que envía el bloque Encoder.
El módulo Self-Attention en el Decoder, reflejado en la Figura C.8, se usa el mecanismo normal, existen variantes, y se muestra cómo se vería la representación de la atención prestada entre sí por cada elemento. Los cuadros mas azules representan que existe una relación contextual fuerte entre dos palabras, por otro lado, los cuadros mas descoloridos muestran la poca relación contextual que tienen dos palabras entre sí.
Al finalizar el bloque Decoder, la salida se pasará a las dos últimas capas del Transformer encargadas de generar las predicciones. La primera genera un vector del tamaño del vocabulario y la segunda es una capa Softmax para ajustar los valores y que sirvan para predecir por ejemplo el siguiente elemento en una secuencia.
En el caso de la generación de texto el elemento dictado por la predicción del Transformer es añadido a la secuencia original incrementando el tamaño de la secuencia a cada nuevo ciclo. Añadir este elemento permite al modelo tener respuestas más detalladas y exactas en relación a lo que ha escrito anteriormente y la secuencia de entrada original.

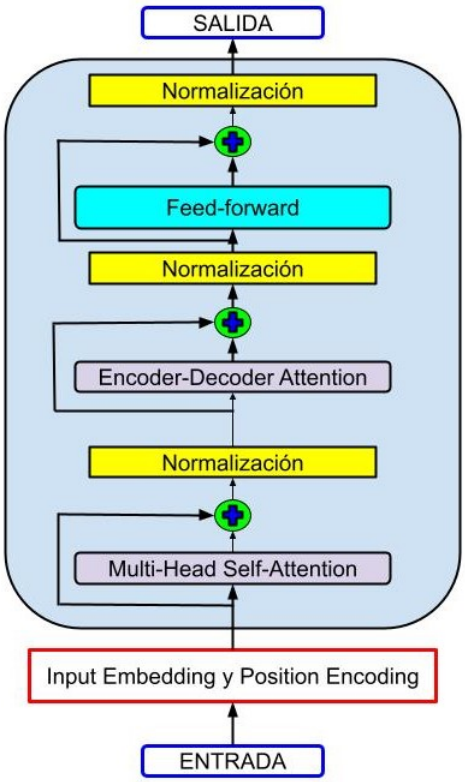

Figura C.7: Contenido interno de un Decoder en el bloque

Cabe destacar que los pesos en cada capa son independientes entre capas del Encoder y Decoder. Otras cosa es que el Decoder recibe como primer elemento un símbolo especial $< start >$ que marca el inicio de la sentencia.
En modelos modernos, estos dos bloques son modificados y en algunos casos descartados en favor de uno, como es el caso de BERT que solo usa los Encoders y GPT que hace lo propio con los Decoders.

C.2. ¿Qué es GPT-2?

Generative Pre-Trained Transformer 2 (GPT-2) es el segundo modelo Transformer desarrollado por OpenAI y fue presentado en Febrero del 2019, es el sucesor de GPT. Está basado en la arquitectura del Transformer tradicional mostrada en la Figura C.4, el primer Transformer se divide en bloques Encoder y Decoder, pero en el caso del modelo GPT, solo usan la parte del Decoder para cumplir su función. Fue desarrollado con el objetivo de predecir el siguiente elemento en una secuencia de forma automática.
GPT-2 es capaz de generar textos completos desde una entrada pequeña, e incluso sin darle entrada es capaz de generar texto coherente que se entiende como texto organizado, de forma tal que tiene relación con el texto de entrada que se da y que las personas son capaces de interpretarlo y comprenderlo, también es capaz de resolver tareas para las cuales no ha sido entrenado, aunque el resultado no siempre es el mejor para este caso.
El nivel de realismo que ha logrado alcanzar este modelo es tal que incluso se ha considerado la situación en la que podrían generarse las famosas Fake News o noticias falsas usando el modelo con el fin de desinformar a la población. Por esa razón se retrasó la liberación de la versión más grande y potente de GPT-2 varios meses desde la salida de las versiones menos capaces.
Es uno de los primeros modelos en superar los mil millones de parámetros, siendo este

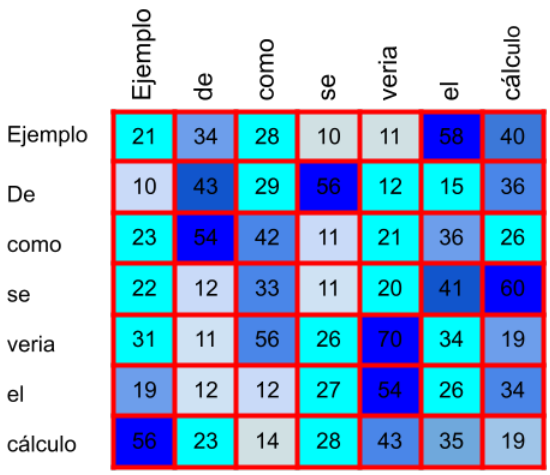

Figura C.8: Matriz que representa la relación entre elementos en el mecanismo Self-Attention

valor muy superior a otros Transformers antes de él, como GPT y BERT. Con GPT-2 comenzó el desarrollo de modelos más grandes con bases de datos de entrenamiento cada vez mayores para reforzar y robustecer estos modelos de lenguaje.

C.2.1. Características

GPT-2 tiene características que lo hacen notar entre otros modelos Transformers, como son:

- Tamaño: 1500 millones de parámetros[1] (1.5 billones en la notación en inglés), en la versión más grande, 10 veces más grande que el primer GPT, y 124 millones en la más pequeña
- Tokenización: Usa Byte Pair Encoding (BPE), que es un nivel de tokenización a nivel sub palabra, por ejemplo la secuencia *El bote azul* se puede dividir en $[El, bo, te, a, zul]$ para representar los elementos en las secuencias.
- Vocabulario: 50257 elementos distintos, palabras, sub palabras, símbolos ortográficos, espacios en blanco, etc.
- Solo usa wl bloque Decoder para procesar las secuencias
- Tamaño del bloque Decoder: 48 Decoders apilados en su versión más grande y 12 en la más pequeña.
- Tamaño máximo para una secuencia: 1024 tokens (elementos) en la secuencia, el doble que en el primer GPT.
- Mecanismo Self-Attention: Usa Masked Self-Attention, un tipo de mecanismo de atención que no busca obtener la relación de una palabra con toda la secuencia sino solo con los elementos a la izquierda, ignora todo lo que se encuentre a su derecha a diferencia de otros modelos.

[1]Se entiende como un parámetro a toda variable que se irá modificando de forma automática durante el entrenamiento de un modelo de Machine Learning, en el caso de las redes neuronales, los pesos asociados y sesgos a las neuronas son considerados parámetros del modelo.

- Zero Shot Learning: Capacidad de resolver una tarea nueva que no se ha presentado durante el entrenamiento, por ejemplo responder preguntas sin haber entrenado el modelo para que lograra esta funcionalidad, solo siguiendo las instrucciones de entrada, debe tomarse en cuenta que los resultados no son los mejores para todos los casos donde se necesita un criterio específico.

C.2.2. Arquitectura

A diferencia del Transformer original dividido por un bloque de Encoders apilados (pila Encoder) y un bloque de bloques Decoders apilados (pila Decoder), GPT-2 solo hace uso del bloque Decoder desechando el bloque Encoder debido al enfoque del modelo. Existen 4 versiones de GPT-2 que están divididas de acuerdo al número de parámetros

- Small: Tiene 117 millones de parámetros y 12 bloques Decoder.
- Medium: Tiene 345 millones de parámetros y 24 bloques Decoder.
- Large: Tiene 762 millones de parámetros y 36 bloques Decoder.
- Extra Large: Tiene 1542 millones de parámetros y 48 bloques Decoder.

El objetivo del GPT-2 es predecir el siguiente token de una secuencia, y la tarea del Decoder es generar un nuevo elemento para una secuencia dada. Por lo cual basta con este bloque para cumplir el objetivo.

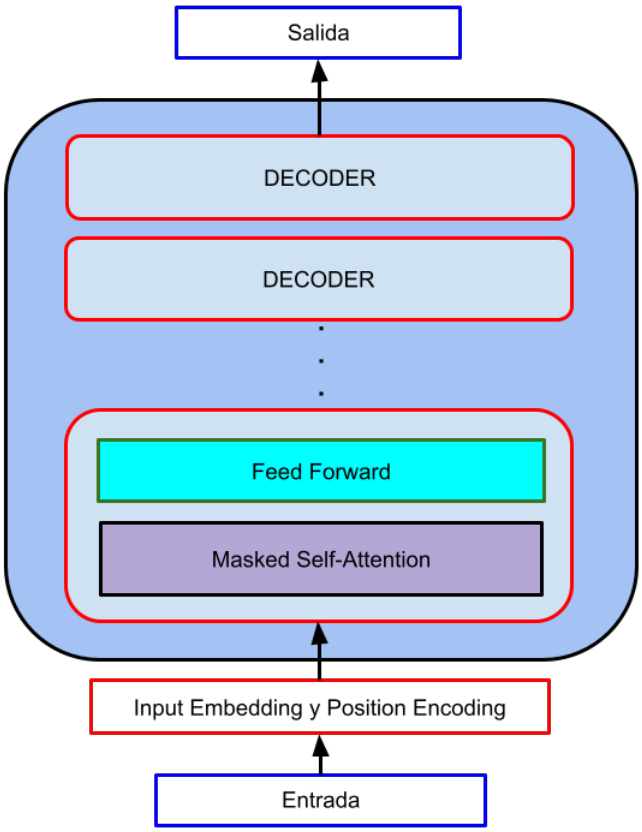

Figura C.9: Apariencia de un decoder en GPT-2

Como se puede ver en la Figura C.9, el bloque Decoder está formada por Decoder conectados. El primero de estos tiene como entrada una secuencia tokenizada gracias al input embedding y al Positional Encoding.

El bloque de entrada procesa la secuencia y genera una salida que pasa a los siguientes bloques hasta alcanzar el último Decoder. La salida se procesa para generar el siguiente elemento de acuerdo a la secuencia de entrada.
Aunque son parecidos al Decoder del Transformer original, los nuevos bloques difieren al usar una variante del mecanismo de "Self-Attention" llamado "Masked Self-Attention".

C.2.3. Masked Self-Attention

La variante del mecanismo de Self-Attention usada en GPT-2 solo mira a los elementos de la secuencia fijando un elemento en específico como pivote. Para el Self-Attention

	Ejemplo	de	como	se	veria	el	cálculo
Ejemplo	21	-inf	-inf	-inf	-inf	-inf	-inf
De	10	43	-inf	-inf	-inf	-inf	-inf
como	23	54	22	-inf	-inf	-inf	-inf
se	22	12	33	11	-inf	-inf	-inf
veria	31	11	56	26	70	-inf	-inf
el	19	12	12	27	54	26	-inf
cálculo	56	23	14	28	43	35	19

Figura C.10: Matriz enmascarada en el módulo Masked Self-Attention

normal cada elemento calcula su relación con todos los demás elementos de una secuencia, no importando si están antes o después.
Pero en el caso de esta variante enmascarada, lo que se hace es cubrir todos los elementos a la derecha del elemento fijo y calcular la relación con los tokens a la izquierda desde este, condicionando a predecir el siguiente token con menos elementos que el proceso normal como se puede ver en la Figura C.10, donde los elementos al lado de la secuencia son ignorados, en este caso se representó con *-inf*.

C.2.4. Byte pair Encoding (BPE)

Este método de codificación a nivel sub palabra es de los más utilizados debido a que reduce enormemente el vocabulario inicial cuando se aplica a grandes conjuntos de datos. Lo que hace este método es agrupar por pares los conjuntos de caracteres más repetidos. El proceso se realiza varias veces según sea necesario.

Ejemplo

Ejemplo adaptado de [28]. Considera el siguiente conjunto de palabras con su respectivo número de repeticiones por palabra

$$Z = [casaca : 3, campo : 3, gato : 2, pato : 2]$$

Se agrega a cada palabra un token de final de palabra "$< /w >$" para identificar dónde acaba cada palabra.
El vocabulario para este conjunto sería

Número	Token	Frecuencia
1	c	9
2	a	16
3	s	3
4	m	3
5	o	7
6	g	2
7	t	4
8	p	5
9	$< /w >$	10

Cuadro C.1: Tabla 1 Ejemplo de BPE

Se identifica el token más frecuente, en este caso “a” y se realiza la combinación más frecuente con otro token en el vocabulario que aparece en Z. que serían: “ca” con 9 apariciones(6 en “casaca” y 3 en “campo”).
Se actualiza la tabla añadiendo el nuevo elemento “ca” y reduciendo las frecuencias en los elementos individuales “c” y “a”.

Número	Token	Frecuencia
1	c	9-2=0
2	a	16-9=7
3	s	3
4	m	3
5	o	7
6	g	2
7	t	4
8	p	5
9	$< /w >$	10
10	ca	9

Cuadro C.2: Tabla 2 Ejemplo de BPE

Ahora se puede tomar el token “$< /w >$” para ayudar a identificar un “ca” a inicio y al final de una palabra, ya que este caso ocurre para el conjunto que se está manejando. se observa que “ca$< /w >$” se repite 3 veces así que se actualiza la tabla

Número	Token	Frecuencia
1	c	9-2=0
2	a	16-9=7
3	s	3
4	m	3
5	o	7
6	g	2
7	t	4
8	p	5
9	$< /w >$	10-3=7
10	ca	9-3=6
11	ca$< /w >$	3

Cuadro C.3: Tabla 3 Ejemplo de BPE

El siguiente paso es usar el token más repetido según la tabla actual, que para este caso es “o” y se concatena junto a su pareja más frecuente en Z, que es “t”. En este caso “to” se repite 4 veces (2 en “gato” y 2 en “pato”).
Se actualiza la tabla

Número	Token	Frecuencia
1	c	9-2=0
2	a	16-9=7
3	s	3
4	m	3
5	o	7-4=3
6	g	2
7	t	4-4=0
8	p	5
9	$< /w >$	10-3=7
10	ca	9-3=6
11	ca$< /w >$	3
12	to	4

Cuadro C.4: Tabla 4 Ejemplo de BPE

Para el siguiente paso se junta “$< /w >$” que en este caso el el token con más frecuencia con “to” para determinar qué es el final de una palabra. “to$< /w >$” aparece 4 veces en el conjunto Z.
Se actualiza la tabla

Número	Token	Frecuencia
1	c	9-2=0
2	a	16-9=7
3	s	3
4	m	3
5	o	7-4=3
6	g	2
7	t	4-4=0
8	p	5
9	$< /w >$	10-3=7 7-4=3
10	ca	9-3=6
11	ca$< /w >$	3
12	to	4-4=0
13	to$< /w >$	4

Cuadro C.5: Tabla 5 Ejemplo de BPE

El vocabulario actual sería

Número	Token	Frecuencia
1	a	7
2	s	3
3	m	3
4	o	3
5	g	2
6	p	5
7	$< /w >$	3
8	ca	6
9	ca$< /w >$	3
10	to$< /w >$	4

Cuadro C.6: Tabla 6 Ejemplo de BPE

Aunque para el ejemplo el vocabulario no se logró reducir debido a que faltan más iteraciones por hacer, con más iteraciones se pueden encontrar los pares “sa”, “po$< /w >$”, “ga”, “pa”, reduciendo el vocabulario de 10 a 8, obsérvese la creación de sub palabras del conjunto Z el cual es uno de los objetivos del método BPE.

Apéndice D

Transfer learning

D.1. Transfer Learning o Transferencia de Conocimiento

En el campo del Machine learning es usual entrenar modelos para resolver tareas en particular como clasificación o traducción. El entrenamiento cuando se hace desde cero es conocido en inglés como *Train from scratch* y tiene altos costos computacionales y la información necesaria para obtener modelos precisos debe ser amplia y verificada para albergar la menor cantidad de elementos que dificulten el aprendizaje.
El entrenamiento no es un inconveniente cuando el modelo es pequeño y necesita pocos datos, ya que el tiempo y poder de procesamiento requeridos no son tan grandes. En cambio, cuando el modelo requiere mucha información para entrenarse, puede llevar una alta demanda de recursos.
En este punto entra la transferencia de conocimiento que consiste en la posibilidad de utilizar modelos previamente entrenados en tareas similares a la objetivo. Los modelos pre entrenados funcionan como base para tareas más especificas.
Entre las ventajas obtenidas usando este enfoque está el uso de menos datos para entrenar el modelo deseado. Otra ventaja es la existencia de una gran cantidad de modelos pre entrenados en millones de datos con alta precisión que son de acceso público para la comunidad y que de otra forma serían imposibles de entrenar desde cero por las dificultades antes mencionadas. Un ejemplo puede ser, el obtener un modelo previamente entrenado en clasificación de imágenes con millones de datos y usarlo de bases de un modelo que busca clasificar cierto tipo de imagen o patrón, como rostros de animales, ecosistemas, entornos, etc.
La transferencia de conocimiento es común llevarla a cabo en modelo grandes siendo los Transformers los elementos idóneos para aplicar este enfoque, donde los Transformers son entrenados en millones de datos para tareas generales dejando espacio a su especialización.

D.1.1. Fine tuning

Fine Tuning (*Ajuste Fino* en español) se puede ver como la realización de ajustes a un modelo pre entrenado para alcanzar un objetivo deseado.
El flujo común que se sigue para ajustar un modelo es simple:

- Obtener un modelo pre entrenado en una tarea similar a la tarea objetivo, en específico los pesos de la red que servirán de base para el modelo.
- Se inicializa el nuevo modelo con los pesos previamente obtenidos.
- Entrenar el modelo nuevo en la base de datos para la tarea objetivo.

Se debe tener en consideración, aunque el tiempo de entrenamiento de los modelos nuevos se reduce drásticamente, la capacidad de cómputo necesaria para cargar y entrenar los

modelos puede ser excesiva. Debido a esto, para el proceso de ajuste fino se suelen usar servicios gratuitos, como Google Collab si el modelo es considerado pequeño y servicios de pago si se requiere el tiempo y poder de cómputo para un modelo grande, por ejemplo AWS, Linode, etc.

D.1.2. Ajustes en Fine Tuning

Existen bastantes formas de ajustar un modelo pre entrenado donde los principales elementos a modificar son las capas de la red y la tasa de aprendizaje.

Entrenar toda la red

El entrenamiento total de la red es la forma más común y sencilla de ajustar un modelo, haciendo que se cambien todos los pesos de la red durante el entrenamiento. También es la forma más tardía para el entrenamiento.

Congelar las capas en el modelo

Permite entrenar capas específicas donde se adquiere información relevante. Normalmente se congelan las primeras capas ya que estas no capturan información tan relevante como capas más profundas en la red. Al final del entrenamiento se descongelan las capas y se añaden al nuevo modelo. El congelar capas permite agilizar el entrenamiento al tener una menor cantidad de valores que modificar.

Transferencia vía tareas intermedias [8]

Consiste en entrenar el modelo pre entrenado en tareas parecidas a la tarea objetivo antes de entrenarlo en la tarea deseada. Para las tareas intermedias se cuenta con bases de datos más grandes que la tarea objetivo. Es ajustar el modelo varias veces, poco a poco, llevando a una mayor necesidad de tiempo y datos para ajustar el modelo a su estado final. Y aún con estos detrimentos, suele dar mejores resultados que los métodos anteriores.

Mixout [9]

Su labor es mantener los valores del modelo pre entrenado original. El proceso se realiza de forma estocástica durante cada iteración donde cada parámetro tiene una probabilidad p de ser cambiado por el valor original del modelo. El objetivo es evitar que el nuevo modelo se separe demasiado del modelo original.

Decaimiento de peso pre entrenado [10]

Es una forma de regularización en la que los parámetros a regular son los pesos de la red. El mecanismo se realiza en cada iteración y se puede ver como

$$\lambda(w_m - w_o)$$

Donde

- λ es un hiper parámetro decidido por el usuario para regular el modelo.
- w_m es el valor del parámetro del modelo durante el entrenamiento.
- w_o es el valor del parámetro del modelo pre entrenado.

Decaimiento de la taza de aprendizaje [11]

Esta forma de ajuste se refiere a controlar el aprendizaje del modelo en sus distintas capas, donde el objetivo es que las capas que capturen la información más relevante tengan un mayor aprendizaje, haciendo su convergencia más lenta que las capas menos importantes.
Como su nombre lo indica, la tasa de aprendizaje se va reduciendo conforme se entrena el modelo. Además de que cada capa tiene una tasa distinta de inicio, siendo las capas más altas una tasa mayor y las más bajas una tasa menor, dejando las capas que capturan la información más básica y general, es decir, las primeras capas del modelo, que aprendan de forma más lenta. Y las capas que aprenden la información más importante para el problema aprendan de forma más rápida, en este caso, las últimas capas del modelo.
También puede usarse agrupando las capas en grupos y dando una taza de aprendizaje específica según el grupo al que pertenezcan reduciendo el número de tasas de aprendizaje necesarias durante el entrenamiento.

Pasos de calentamiento

Es una variación del método anterior donde la tasa de aprendizaje inicia desde cero y tiene que llegar a un valor específico es un número determinado de pasos para luego ir disminuyendo como en el método anterior. En otras palabras es ir

$$0 \rightarrow X \rightarrow 0$$

donde X es la tasa de aprendizaje específica para esa capa o grupo de capas.

Reiniciar capas de la red

Parecido al primer método mostrado solo que el objetivo es reiniciar el valor de las capas más cruciales para el problema en lugar de ajustar los valores pre entrenados, estas son normalmente las capas donde se aprende la información específica necesaria para resolver el problema, siempre son las últimas capas.
Debe de tomarse en cuenta el número de capas a reiniciar ya que esto podría afectar el resultado final si se borran muchas debido al gran cambio con respecto al modelo original o en el caso contrario, si se reinician pocas capas el cambio no será tan significativo.

Promedio de pesos estocástico [12]

La forma de trabajo en este método es obtener los valores del modelo final promediando los valores de la red durante cada etapa en el entrenamiento con respecto a la etapa anterior y al número de modelos calculados.
El proceso de entrenamiento se divide en dos partes, asignando a cada parte un porcentaje del tiempo de entrenamiento total:

- Primero se entrena el modelo durante un porcentaje específico de épocas y al final de este primer entrenamiento se guardan los pesos de la red actual. En esta parte se suele usar un calendario para designar la tasa de aprendizaje.
- Para el porcentaje restante de épocas, se entrena desde el modelo guardado previamente pero con una tasa de aprendizaje fija (o cíclica). Para cada época se captura el modelo actual y se promedian los valores de los modelos para al final obtener los valores finales del modelo.

Apéndice E

Poesía

E.1. Elementos de la Poesía

E.1.1. Prosa

La Prosa es la escritura común sujeta solamente a las reglas gramaticales cotidianas. Usada para expresar conceptos, ideas hechos tal cual pensamos y hablamos.
La prosa como parte de la poesía se conoce como prosa literaria, y usa el sentido lírico (actitud lírica) de la poesía para expresar sus ideas, pero sin estar sujeta a la rima y la métrica, además, de no perder su naturalidad. El texto escrito en prosa no cambia de renglón a menos que se quiera expresar una nueva idea distinta a la actual.

E.1.2. Verso

El verso al contrario de la prosa se rige de reglas estrictas para preservar la belleza más allá de la combinación de palabras, sino también de la estructura. Se caracteriza por la separación de los renglones en frases cortas formando estrofas.
Los versos se pueden dividir de acuerdo a ciertos aspectos:

- Según la métrica: Número de sílabas en cada verso.
- Según la rima: Coincidencia en la última vocal acentuada y la última sílaba del texto.
- Verso libre: no se apega a ninguna de las dos reglas anteriores, tampoco se aplica un efecto lírico en sus versos. Siendo la diferencia con la prosa lírica.

E.1.3. Métrica

La métrica se puede entender como el número de sílabas que existen en cada verso. Los versos clasificados por la métrica se pueden separar de acuerdo al número de sílabas por renglón en dos tipos:

- Verso de arte menor: Entre 2 y 8 sílabas.
- Verso de arte mayor: Desde nueve sílabas.

Se debe especificar que se refieren a sílabas poéticas. Al contar las sílabas en un texto poético se siguen algunas reglas especiales, en español se usa la "ley de mussafia.º último acento del verso, según el tipo de la última palabra del verso se sumará o restará una sílaba al verso. Esta regla se divide en tres casos:

- Cuando un verso es oxítono: se suma una sílaba.
- Cuando el verso es paroxítono: no se resta ni suma sílabas.
- Cuando el texto es proparoxítono: se resta una sílaba.

E.1.4. Licencia Poética

La licencia poética es el recurso literario encargado de mantener o modificar el número de sílabas dentro de un verso de acuerdo a las necesidades del texto. Existen 3 tipos principales:

- Sinalefa: Unir dos vocales en una sola sílaba, cuando una termina en vocal y la otra en vocal o "h".
- Dialefa: Se aplica en diptongos y los separa en dos sílabas distintas.
- Sinéresis: Unión en una sílaba de dos vocales fuertes (a,e,o) pertenecientes a distintas sílabas.

E.1.5. Rima

La rima es la similitud fonética, o sonido de las lenguas, entre palabras de distintos versos en una misma estrofa. Se busca una armonía acústica con palabras que sean de timbre similar.
De acuerdo al timbre se clasifican en:

- Consonante: O rima perfecta, se refiere a que a partir de la última sílaba acentuada del verso, las consonantes y vocales deben ser iguales.
- Asonante: O rima imperfecta, ya que solo es necesario que se repita la última vocal de la última palabra acentuada del verso sin importar el efecto fonético. Permitiendo más libertad en la selección de las palabras posibles para usar.

Considerando el orden de los versos que riman en la estrofa también se puede hacer la siguiente clasificación:

- Rima continua: Tiene la forma $AAAA$. Los versos tienen el mismo timbre.
- Rima gemela o pareada: Tiene la forma AA. Conocidas como coplas, son estrofas de dos versos.
- Rima abrazada: Tiene forma $ABBA$. Donde dos versos que riman quedan dentro de otros dos que igual riman pero con distinto timbre.
- Rima cruzada: Tiene forma $ABAB$. Los versos con mismo timbre se cruzan entre sí.
- Rima encadenada: Es una extensión de la rima cruzada para estrofas con más de 4 versos.

E.1.6. Figuras Literarias

Una figura literaria busca expresar las ideas de una forma más bella haciendo uso del vocabulario. Existe una gran cantidad de figuras literarias que se usan en la literatura. Las principales son:

- Metáfora: Hace uso de conceptos similares a la que se quiere hacer referencia. Por ejemplo, "Tu mirada es tan profunda como el oceano".
- Personificación o humanización: Dar propiedades humanas a objetos inanimados, ideas o animales. Por ejemplo, "El sol me sonreía a cada paso del camino".
- Alegoría: Referirse a una idea de forma indirecta a través de una sucesión de metáforas. Por ejemplo, "La vida es un libro que recoge palabras a cada paso que das y sus páginas hablarán de cómo la has vivido".

- Hipérbole: Aumentar o disminuir un aspecto o acción de forma exagerada. Por ejemplo, "Hace tanto calor, que me estoy derritiendo en la sombra".
- Oxímoron: Generar contradicciones haciendo uso de términos o expresiones contrarias. Por ejemplo, "Las sombras brillantes que iluminan tus ojos vacios".
- Onomatopeya: Representación del sonido mediante palabras. Por ejemplo, "¡PUM!, ¡ROAR!, Kikiriki, cuac-cuac, toc toc toc, zzzz".

E.1.7. Géneros poéticos principales

La poesía se puede dividir en tres grandes grupos de acuerdo a lo que busca expresar:

- Poesía lírica: Busca expresar sentimientos, ideas y emociones apoyándose del lenguaje. Sus subgéneros son la oda, la sátira e incluso las canciones y los himnos. Están apoyados en gran medida del efecto fonético que pueden generar mediante el timbre en sus letras.
- Poesía épica: Escrita normalmente en verso, busca expresar eventos extraordinarios, ya sea reales o no. Se intenta estructurar de tal forma que los eventos históricos coincidan con los relatos. Sus subgéneros son los cuentos, las epopeyas, las fábulas y las leyendas. Por ejemplo la odisea, la Iliada, los siete contra Tebas, etc.
- Poesía dramática: Combinación de las dos anteriores centrada en la actuación en teatros, ya sea obras o musicales. Se hace uso de cantos y coros para apoyar el efecto poético. Uno de sus subgéneros son la ópera y la tragedia. Por ejemplo, El fantasma de la opera, Mozart'l Opera Rock, etc.

Apéndice F

Encuestas

F.1. Cuestionario

1. ¿Te gusta el siguiente poema?

   A veces en la noche, cuando la luna brilla,
   la luz de la vida se apaga
   y el amor se desvaneces
   en el fondo de un lago de agua.
   ¡Oh,amor!
   ¿Adonde,oh amor?
   En mi alma, como en mi corazón
   no hay nada que no sea tuyo
   no hay nadie que me olvide de ti,
   ni siquiera tu nombre.
   Tu voz es como el río que se pasa por mi garganta.

   - A) SI
   - B) NO

2. ¿Qué poema te gusta más?

Opción 1	Opción 2
¡Oh, que es el amor!¡Que es la vida! ¿Qué es eso?¿Cómo es esto? La vida es una cosa que no se sabe que se pierde en la noche, en el viento y que nunca se ha vuelto a ver, y no puede se otra cosa más que un sueño, una herida o una flor, o un beso, un temblor, nada más ni más ni menos que una estrella ni nada menos.	Región de manos sucias de pinceles sin pelo de niños boca abajo de cepillos de dientes Zona donde la rata se ennoblece y hay banderas innúmeras y cantan himnos y alguien te prende, hijo de puta, una medalla sobre el pecho Y te pudres lo mismo.

   - A) Opción 1
   - B) Opción 2

3. ¿Te gusta el siguiente poema?

   En la noche de mayo
   y en la madrugada de junio
   en el huerto de los muertos
   de las almas que se van a casar
   con la luna y con el sol, con las rosas
   del cielo y la tierra, a la luz del sol
   que alumbra el día, al sol del día
   como una rosa, como un beso de luna
   sobre el cuerpo de un hombre que no sabe
   que hacer, que decir, como decir
   lo que dice el mundo:¿Qué hacer?

   - A) SI
   - B) NO

4. ¿Qué poema te gusta más?

   La noche
   es tierra,
   y la tierra
   no es nada
   sino una isla
   en la que el viento
   se detiene
   para siempre
   como un río
   o una montaña
   que se aleja
   de las aguas
   a la orilla del mar
   donde la aurora
   pasa por el mundo
   con las alas abiertas.

   Opción 1

   La guitarra,
   hace llorar a los sueños.
   El sollozo de las almas
   perdidas,
   se escapa por su boca
   redonda.
   Y como la tarántula
   teje una gran estrella
   para cazar suspiros,
   que flotan en su negro
   aljibe de madera.

   Opción 2

   - A) Opción 1
   - B) Opción 2

Casa abierta al tiempo
UNIVERSIDAD AUTÓNOMA METROPOLITANA

ACTA DE EXAMEN DE GRADO

No. 00099
Matrícula: 2202800399

Generación Automatizada de Textos: Poesía

Anthony

ANTHONY PEREZ RANGEL
ALUMNO

REVISÓ

MTRA. ROSALÍA SERRANO DE LA PAZ
DIRECTORA DE SISTEMAS ESCOLARES

En la Ciudad de México, se presentaron a las 11:00 horas del día 19 del mes de septiembre del año 2022 en la Unidad Iztapalapa de la Universidad Autónoma Metropolitana, los suscritos miembros del jurado:

DR. SERGIO GERARDO DE LOS COBOS SILVA
DRA. MARIA ELENA LARRAGA RAMIREZ
DR. ERIC ALFREDO RINCON GARCIA

Bajo la Presidencia del primero y con carácter de Secretario el último, se reunieron para proceder al Examen de Grado cuya denominación aparece al margen, para la obtención del grado de:

MAESTRO EN CIENCIAS (CIENCIAS Y TECNOLOGIAS DE LA INFORMACION)

DE: ANTHONY PEREZ RANGEL

y de acuerdo con el artículo 78 fracción III del Reglamento de Estudios Superiores de la Universidad Autónoma Metropolitana, los miembros del jurado resolvieron:

APROBAR

Acto continuo, el presidente del jurado comunicó al interesado el resultado de la evaluación y, en caso aprobatorio, le fue tomada la protesta.

DIRECTOR DE LA DIVISIÓN DE CBI

Román Linares Romero
DR. ROMAN LINARES ROMERO

PRESIDENTE

DR. SERGIO GERARDO DE LOS COBOS SILVA

VOCAL

DRA. MARIA ELENA LARRAGA RAMIREZ

SECRETARIO

DR. ERIC ALFREDO RINCON GARCIA