UNIVERSIDAD NACIONAL DE INGENIERÍA

ESCUELA DE CIENCIA DE LA COMPUTACIÓN

Seminario de tesis 2:

Sistemas de Preguntas y Respuestas con redes LSTM, Mecanismos de Atención y modelos pre-entrenados

AUTOR

José Luis Navío Torres

ASESOR

Prof. César Jesús Lara Ávila

LIMA-PERÚ
2021

…

Agradecimientos

A mi madre que me apoya en todos mis proyectos.
Al profesor César Lara Ávila que me apoyó con su conocimiento, lecturas, comentarios y sugerencias.
A mis compañeros de ACECOM que me motivaron a investigar temáticas de NLP y al desarrollo del proyecto.

Índice de figuras

Índice general

Generalidades

1.1. Problemática

Los modelos de comprensión lectora tienen un gran valor práctico, especialmente para fines de la industria, ya que un modelo debidamente entrenado puede aplicarse a un sistema de chatbot, cuya función es responder preguntas frecuentes sin necesidad de la intervención humana. En los últimos años se han desarrollado importantes avances gracias a la aplicación de la Inteligencia Artificial en este tipo de tareas, con el desarrollo de modelos que funcionan con un gran desempeño para el tratamiento de textos en el lenguaje inglés, sin embargo, para el idioma español, no existen la misma cantidad de estudios y modelos desarrollados para estas mismas tareas.
Con la evolución de las técnicas de la Inteligencia Artificial se han desarrollado una gran variedad de modelos basados en redes neuronales tradicionales que ofrecen resultados aceptables, posteriormente se desarrollaron modelos basados en mecanismos de atención que mejoraron las métricas en determinadas tareas y en los últimos años se alcanzó el estado del arte para muchas tareas de NLP con el uso arquitecturas de mecanismos de atención que utilizan una arquitectura de codificador-decodificador y computo paralelo conocidos como Transformers. A pesar de que estos modelos basados en Transformers brinden las mejores métricas para las tareas de NLP, sus tamaños suelen ser grandes, lo cual dificulta su despliegue para producción.
En ese contexto se describe la pregunta que motiva la presente investigación: ¿Cuál es la arquitectura de red neuronal que nos ofrece mejores resultados en métricas y tiempos de inferencia para la implementación en la industria de un sistema de preguntas y respuestas en lenguaje español ?

1.2. Objetivos

Los objetivos para el presente proyecto son:

1.2.1. Objetivo General

Implementar arquitecturas de redes neuronales para la tarea de Procesamiento de Lenguaje Natural (NLP) de Preguntas y Respuestas, y realizar un estudio de sus métricas para la implementación en una etapa de despliegue.

1.2.2. Objetivos Específicos

O1) Implementar y evaluar modelos basados en mecanismos de atención y redes LSTM: DrQA.

O2) Implementar y evaluar modelos basados en la arquitectura de Transformers como son BERT, RoBERTa y ELECTRA.

O3) Realizar un cuadro comparativo e identificar el modelo que proporcione mejores métricas para su uso en aplicaciones prácticas.

O4) Desarrollar una implementación de un prototipo de despliegue con el framework Streamlit.

1.3. Herramientas y Métodos

1.3.1. Herramientas

Las principales a herramientas utilizar serán:

1. Para el desarrollo de los modelos, el framework de código abierto Pytorch. https://pytorch.org/
2. Para el entrenamiento, una máquina virtual de Colab con una GPU NVIDIA Tesla T4 de 16 GB de RAM.
3. Para el prototipo de despliegue, una laptop Core I7 con 8 Gb de RAM.

1.3.2. Metodología

En base a los objetivos específicos mencionados, las actividades a desarrollar serán:

O1) Entrenar un modelo base para las comparativas de los modelos, en primer lugar se desarrollará una arquitectura basada en redes neuronales recurrentes.

O2.1) Entrenar los modelos BERT y su variante RoBERTa en español con la técnica de Transferencia de Aprendizaje para la tarea de Preguntas y Respuestas. Se utilizarán las métricas F1 y Exact Match para medir el desempeño del modelo.

O2.2) Entrenar el modelo de ELECTRA en español con la técnica de Transferencia de Aprendizaje para la tarea de Preguntas y Respuestas. Se utilizarán las métricas F1 y Exact Match para medir el desempeño del modelo.

O3) Establecer cual es el modelo que nos brinde mejores resultados, considerando las métricas estudiadas y el tiempo de respuesta ante nueva data.

O4) Desarrollar la implementación del sistema de Preguntas y Respuestas con el framework de streamlit, con los modelos que hayan mostrado mejores resultados.

1.4. Alcances y Limitaciones

1.4.1. Hipótesis

Es posible determinar la arquitectura de Red Neuronal que brinde las mejores características de despliegue para una aplicación de un sistema de preguntas y respuestas, basado en el análisis y comparativa de arquitecturas de mecanismos de atención con LSTM y arquitecturas de tipo Transformers con modelos pre-entrenados para el lenguaje español.

1.4.2. Justificación

La presente investigación se justifica por su uso práctico debido a que reconocer la arquitectura de red neuronal con mejor desempeño para el despliegue de un sistema de preguntas y respuestas nos permitirá implementar futuros proyectos más elaborados como sistemas de preguntas y respuestas abiertas o chatbots que son sistemas con mucha aplicación para la industria.

Marco Conceptual

2.1. Procesamiento de Lenguaje Natural

El Procesamiento de Lenguaje Natural (NLP, por sus siglas en inglés), es una rama de la Inteligencia Artificial que estudia el uso de las computadoras para leer, entender y procesar el lenguaje humano.
Las tareas de NLP se pueden dividir en Generación de Lenguaje Natural (NLG, por sus siglas en inglés) y Comprensión de Lenguaje Natural (NLU, por sus siglas en inglés).
Las tareas dentro de NLU son las de Recuperación de Información, Traducción de Máquina, Sistema de Preguntas y Respuestas, Clasificación de Texto y Sumarización de Texto [1].

2.2. Redes Neuronales

Son un conjunto de neuronas computacionales que han sido entrenadas para procesar información y desarrollar acciones específicas de una tarea. Las neuronas están conectadas a través de conexiones con pesos, las que transmiten información hacia las demás neuronas.
Una red neuronal está compuesta, en su mayoría, por una capa de entrada, una o más capas ocultas y una capa de salida.
La capa de entrada se encarga de recibir la información procesada como un vector de números, que pueden ser la representación de una imagen, un texto, un audio, un vídeo, etc. Las capas ocultas se encargan de generar las características específicas de una tarea cómo encontrar patrones en los datos de entrada. La capa de salida se encarga de recopilar toda la información de la última capa oculta y mostrar los resultados como salida [2].

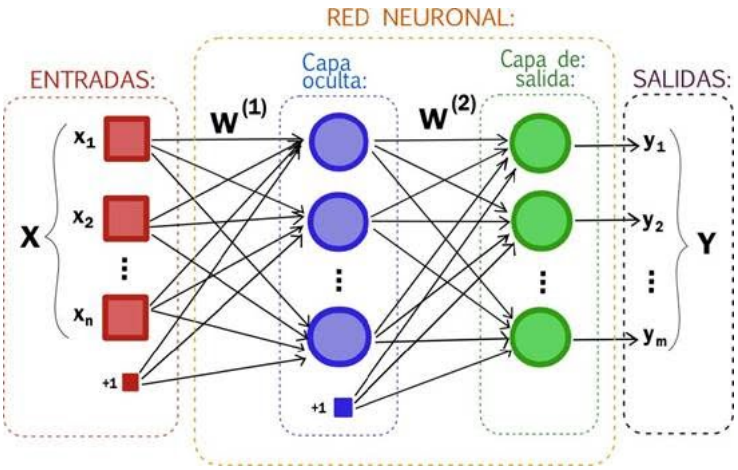

Figura 2.1: Red Neuronal de 3 capas: Capa de entrada, Capa Oculta y Capa de Salida [3]

Las redes neuronales son usadas para resolver problemas complejos no lineales. Siendo la función de activación la que introduce una no linealidad al sistema. Entre estas funciones de activación

son conocidas las funciones Sigmoidea, ReLU y Tanh. Los tipos de redes neuronales son las redes neuronales totalmente conectadas, las redes neuronales convolucionales y las redes neuronales recurrentes [4].

2.2.1. Redes Neuronales Recurrentes

Las redes neuronales recurrentes (RNN, por sus siglas en inglés) poseen una estructura de redes neuronales que procesan entradas de longitud variable. El diseño básico consiste en aplicar la misma red 'S' para cada elemento de entrada y transmitir la información entre elementos adyacentes. La ventaja de este enfoque consiste en reducir el número de parámetros y permitir el procesamiento de una secuencia de entrada de cualquier longitud, lo cual las hace adecuadas para las tareas de NLP como Traducción de Máquina, Sistema de Preguntas y Respuestas, Clasificación de Texto entre otras [5].

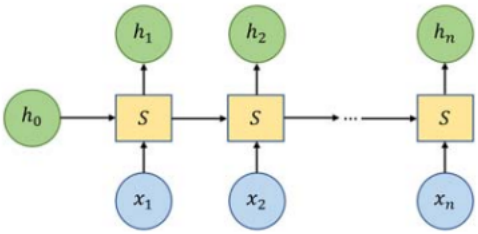

Figura 2.2: Red Neuronal Recurrente [5]

La imagen muestra la arquitectura de una RNN donde la entrada está representada por $\{x_1, x_2, ..., x_n\}$, y la información entre los elementos es transmitida por los vectores de estado ocultos $\{h_0, h_1, ..., h_n\}$. Se puede observar que un estado oculto h_t posee información de la palabra x_t y de las palabras anteriores.
El problema principal de las redes neuronales recurrentes es el desvanecimiento de gradiente o la explosión de gradiente, lo que las impide trabajar bien ante secuencias largas. Por ese motivo se diseñaron modificaciones utilizando puertas, una de estas variantes es conocida como LSTM [6].

2.2.1.1. Memoria larga a corto plazo (LSTM)

LSTM es una de las formas más usadas de redes neuronales recurrentes, ya que son capaces de aprender dependencias en largas secuencias de entrada. La diferencia en el funcionamiento de un LSTM es que cada celda 'S' puede escoger si la información es almacenada u olvidada, debido a que contiene puertas. Éstas puertas son parte de una red neuronal distinta que aprende que palabras dejar pasar u olvidar para su posterior procesamiento [6].

Las ecuaciones que describen el comportamiento de las puertas de la celda LSTM mostrada en la Figura 2.3 son:

- **Puerta de entrada**:

  La puerta de entrada decide qué parte de la nueva información se almacenará en la memoria a largo plazo. Funciona con operaciones a la información de la entrada actual y la memoria a corto plazo del paso de tiempo anterior.

$$i_1 = \sigma(W_{i_1} \cdot (H_{t-1}, x_t) + bias_{i_1})$$

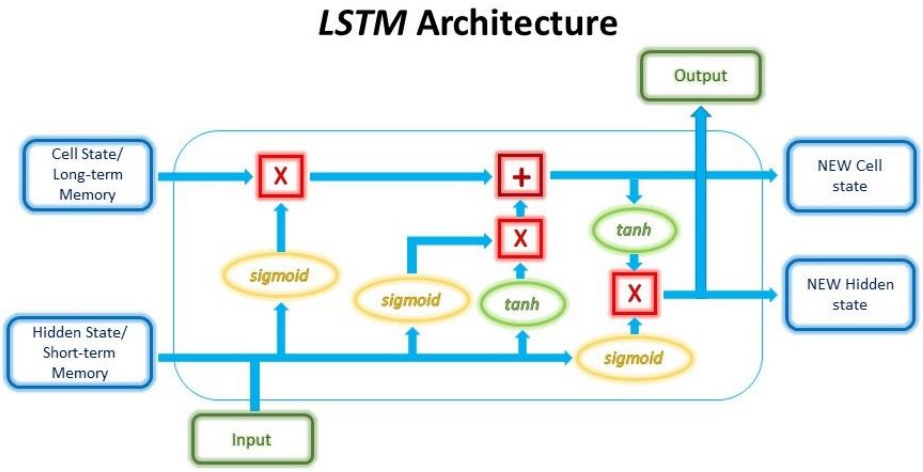

Figura 2.3: Arquitectura de celda LSTM [7]

$$i_2 = tanh(W_{i_2} \cdot (H_{t-1}, x_t) + bias_{i_2})$$
$$i_{input} = i_1 * i_2$$

- **Puerta de olvido**:

  La puerta del olvido decide qué información de la memoria a largo plazo debe conservarse o descartarse. Esto se hace multiplicando la memoria entrante a largo plazo por un vector de olvido generado por la entrada actual y la memoria entrante a corto plazo.

$$f = \sigma(W_{forget} \cdot (H_{t-1}, x_t) + bias_{forget})$$
$$C_t = C_{t-1} * f + i_{input}$$

- **Puerta de salida**:

  La puerta de salida tomará la entrada actual, la memoria a corto plazo anterior y la memoria a largo plazo recién calculada para producir la nueva memoria a corto plazo que se transmitirá a la celda en el siguiente paso de tiempo.

$$O_1 = \sigma(W_{output_1} \cdot (H_{t-1}, x_t) + bias_{output_1})$$
$$O_2 = tanh(W_{output_2} \cdot C_t + bias_{output_2})$$
$$H_t, O_t = O_1 * O_2$$

2.2.2. Arquitecturas Codificador-Decodificador

La arquitectura Codificador-Decodificador, en su forma básica, está constituida por 2 redes neuronales recurrentes, la primera para la codificación de la secuencia de entrada en un único vector y la segunda que transforma este vector en una secuencia de salida. El codificador internamente usualmente está compuesto por redes tipo LSTM [8] .
La arquitectura codificador-decodificador es útil para varias aplicaciones de tipo secuencia a secuencia como son: Traducción de máquina, Sumarización de texto, Chatbots, Subtítulos de imagen.

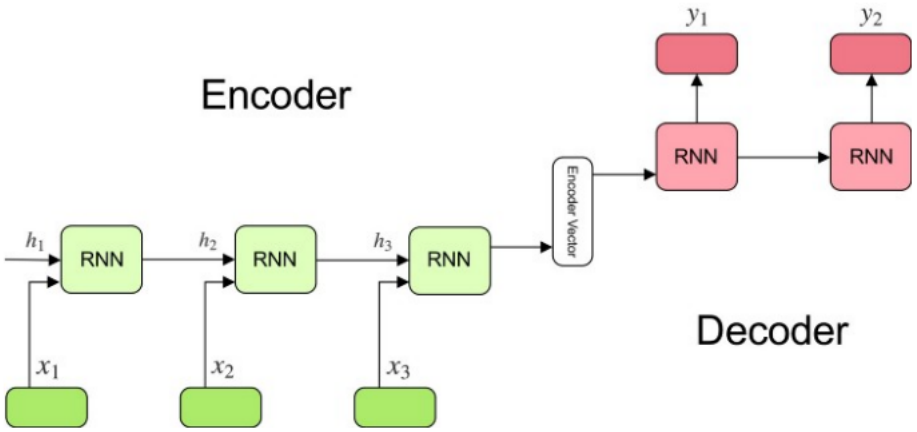

Figura 2.4: Arquitectura Codificador-Decodificador [8]

2.2.3. Mecanismos de Atención

Los mecanismos de atención nacieron con la intención de resolver el problema de las arquitecturas Codificador-Decodificador basadas en redes neuronales recurrentes, las cuales pierden información de las primeras palabras de una secuencia al procesar una entrada muy larga. La entrada a un mecanismo de atención incluye:

1. Los items a atender, que son representados por un grupo de vectores. : $\{a_1, a_2, ..., a_n\}$;
2. El item que presta atención a los items anteriores, representado por un solo vector: x

Primero, se computa la fuerza de la relación entre x y $\{a_1, a_2, ..., a_n\}$, resultando un vector de n puntuaciones, donde una alta puntuación indica una fuerte relación. Esto se implementa a través de una función de atención que produce una puntuación de la similaridad de 2 vectores, esta función puede ser de tipo: producto punto, aditiva, basado en ubicación, etc. (para ver otras variantes de mecanismos de atención consultar el enlace de Lilian Weng [9]).

$$s_i = f(x, a_i) = x^T a_i$$

Luego una función softmax normaliza las puntuaciones para conseguir los pesos de atención.

$$\beta_{i,j} = \frac{e^{s_i}}{\sum_{j=1}^{n} e^{s_j}}$$

Finalmente, los pesos son empleados para computar una suma ponderada de los items atendidos, resultando un vector de atención. Entonces un vector de atención se podría definir como una combinación lineal de los items a ser atendidos $\{a_1, a_2, ..., a_n\}$ y los pesos normalizados provenientes de la interacción entre el item que presta atención x y cada elemento atendido a_i.

$$c_j = \beta_{j,1} a_1 + \beta_{j,2} a_2 + ... + \beta_{j,n} a_n$$

Una importante aplicación del mecanismo de atención se da en el decodificador, ya que éste necesita enfocarse en ciertas partes de la entrada de texto cuando genera nuevas palabras. El codificador usa un RNN para procesar el texto de entrada y su último estado oculto es usado como el estado inicial oculto del decodificador.
Considerando el estado oculto de las palabras de entrada : $\{a_1, a_2, ..., a_n\}$, cuando el decodificador

intenta usar el estado h_{t-1} para generar la *tesima* palabra, éste atiende a los estados ocultos del texto de entrada y computa un vector de atención c_t [5]:

$$Attention((a_1, a_2, ..., a_n), h_{t-1}) = c_t$$

El vector de atención es conocido como **vector de contexto**. Este vector de contexto es unido con el estado oculto h_{t-1} y es introducido al RNN. De esta forma el decodificador puede enfocarse en diferentes partes del texto de entrada en cada paso de decodificación.

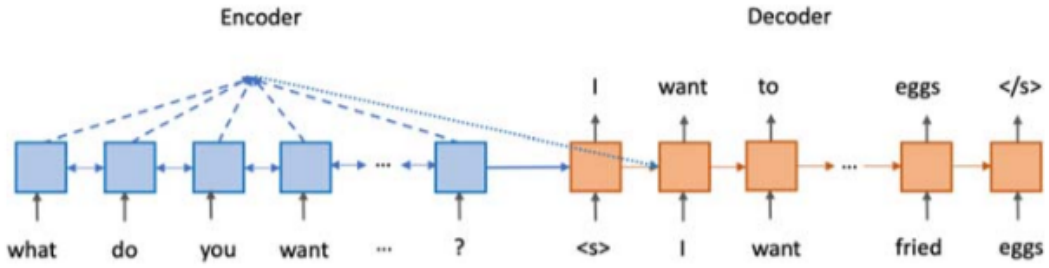

Figura 2.5: Arquitectura codificador-decodificador con un mecanismo de atención [5]

En lugar de construir una secuencia de salida a partir de un único vector de contexto que contenga toda la información del codificador en la última capa oculta, un mecanismo de atención computa un vector de contexto que enlaza cada entrada del codificador y el estado actual a predecir del decodificador.[9].

Los tipos de mecanismos de atención utilizados en los modelos con arquitecturas de redes neuronales para la tarea de NLP de tipo comprensión de lectura de máquina son: Atención cruzada y Auto-atención [5].

2.2.3.1. Atención cruzada

Para el caso de la tarea de NLP de Preguntas y Respuestas, se tienen como entrada 2 bloques de texto que son el artículo y la pregunta. Sobre estas entradas se aplica el mecanismo de atención para procesar su relación semántica [5].

En la figura se considera como el vector de palabras de artículos a $\{p_1, p_2, ..., p_m\}$ y al vector de palabras de preguntas a $\{q_1, q_2, ..., q_n\}$ Una función de atención se aplica para obtener la similaridad entre los vectores $\{p_i\}$ y $\{q_j\}$, cómo por ejemplo el producto interno:

$$s_{i,j} = p_i^T q_j$$

Luego se aplica una función de softmax para normalizar el resultado:

$$\beta_{i,j} = \frac{e^{s_{i,j}}}{\sum_{k=1}^{n} e^{s_{i,k}}}$$

Finalmente se obtiene el vector de atención $\{p_i^q\}$ que es la suma ponderada sobre el vector $\{q_1, q_2, ..., q_n\}$:

$$p_i^q = \beta_{i,1} q_1 + \beta_{i,2} q_2 + ... + \beta_{i,n} q_n$$

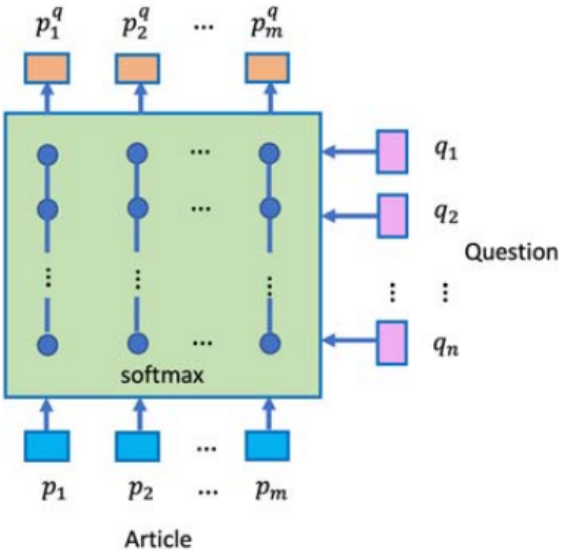

Figura 2.6: Mecanismo de atención cruzada sobre la tarea de preguntas y respuestas [5]

Se puede denotar el proceso de una función de atención como:

$$Attention((p_1, p_2, ..., p_m), (q_1, q_2, ..., q_n)) = (p_1^q, p_2^q, ..., p_m^q)$$

Similarmente, se puede computar el vector de atención desde las preguntas hacia los artículos, que es q_i^p. De esta forma el modelo comprenderá la semántica de cada palabra de las preguntas basada en su relación con el artículo.

2.2.3.2. Auto-atención

Las RNN transfieren información de manera lineal, lo que ocasiona que muy poca información de las primeras palabras sea transmitida a las últimas mientras el artículo sea más largo. Sin embargo, algunas respuestas requieren la comprensión de distintas partes de un artículo. Para resolver este problema se pueden utilizar mecanismos de auto-atención que son parecidos a los de atención cruzada. Los mecanismos de auto-atención computan los vectores de atención desde un grupo de vectores sobre mismos, donde se calculan las puntuaciones de todos los pares de palabras independientemente de su ubicación en el texto.

$$Attention((p_1, p_2, ..., p_m), (p_1, p_2, ..., p_m)) = (p_1^{self}, p_2^{self}, ..., p_m^{self})$$

Esto permite que la información entre palabras a cualquier distancia pueda interactuar directamente, mejorando la eficiencia de la red neuronal. Además, desde que la puntuación de atención sobre cada palabra es independiente, la computación paralela puede usarse para acelerar el proceso. Sin embargo, los mecanismos de auto-atención descartan la información posicional de las palabras, la cual es importante para la comprensión de un lenguaje. Para resolver este problema se utilizan técnicas cómo agregar posicionamiento codificado para agregar la información posicional dentro de la auto-atención. [5]

2.2.4. Modelos Transformer

Una RNN está limitada por su estilo de computación secuencial, por lo que difícilmente se puede realizar paralelización sobre ellas. Por ese motivo se desarrollaron los modelos Transformer [10], los cuales usan mecanismos de atención multicabezal, codificación de posición, normalización de capas y redes neuronales totalmente conectadas.

Externamente posee una estructura de Codificador-Decodificador, donde bloques de codificadores y decodificadores se apilan unos sobre otros. Las capas apiladas son idénticas, además cada capa contiene dos subcapas: la primera es un mecanismo de atención multicabezal y la segunda es una red neuronal totalmente conectada.

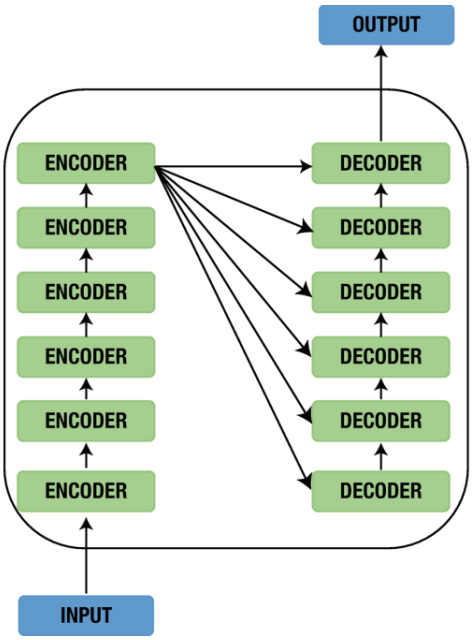

Figura 2.7: Arquitectura externa de un transformer [2]

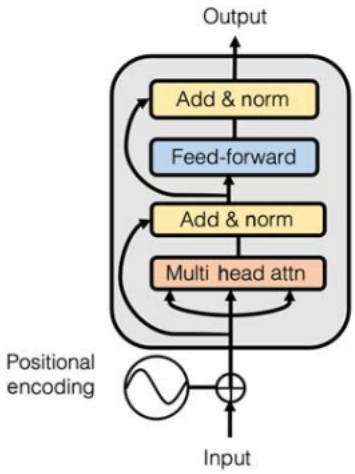

Figura 2.8: Arquitectura interna de un transformer [5]

2.2.4.1. Atención multicabezal

El mecanismo de atención calcula las puntuaciones más relevantes entre una palabra, definida como query (Q), y una secuencia de palabras, definida como keys(K), luego se calcula una puntuación normalizada para computar la suma ponderada de una secuencia de palabras, definida como values(V) [5].

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_{model}}})V$$

La atención multicabezal es la repetición de este proceso h veces para luego concatenar los resultados. Considerando W como la matriz de parámetros que relaciona los vectores Q, K, V, se define la función cabezal cómo:

$$head_i = Attention(QW_i^Q, KQ_i^K, VW_i^V)$$

Entonces la función multicabezal se define cómo:

$$Multihead(Q, K, V) = Concatenation(head_1, ..., head_h)W^O$$

2.2.4.2. Codificación posicional

El Transformer aplica codificación posicional para preservar la información del orden de las palabras, ya que el orden es importante para entender la semántica de un texto. En la codificación posicional se asigna cada posición a una único vector de dimensión d_{model}. Por ejemplo utilizando una función trigonométrica para la codificación posicional podría ser:

$$PE_{pos,2i} = sin(pos/10000^{2i/d_{model}})$$

$$PE_{pos,2i+1} = cos(pos/10000^{2i/d_{model}})$$

Donde $PE_{pos,j}$ representa el valor del j_{th} componente del vector de la pos_{th} posición [2].

2.2.4.3. Capa de normalización

Nos ayuda a evitar el problema de sobresensibilidad en las capas altas, que genera un entrenamiento inestable. Para ello el transformer utiliza una capa de normalización que actúa sobre la entrada de cada capa, aplicando la media y desviación estandár. Considerando x como el vector de entrada y d su dimensión:

$$\mu = \frac{1}{d}\sum_{i=1}^{d} x_i, \sigma = \sqrt{\frac{1}{d}\sum_{i=1}^{d}(x_i - \mu)^2}$$

2.2.4.4. Red totalmente conectada

El transformer contiene una red totalmente conectada que comprende 2 capas lineales y una función de activación ReLU. Considerando x como el vector de entrada y los parámetros W, como la matriz de pesos y b como el bias tenemos:

$$FFN(x) = ReLU(xW_1 + b_1)W_2 + b_2$$

2.3. Modelos Pre-Entrenados

Los modelos pre-entrenados fueron primero aplicados en el área de visión por computadora, siendo AlexNet en el 2012, el modelo de Aprendizaje Profundo que obtuvo el primer lugar en una competencia de reconocimiento de imágenes ImageNet. Los modelos de lenguaje pre-entrenados han sido aplicados en NLP desde el 2015 con la aparición de CoVe [11], ELMo [12], GPT [13], BERT [14] y recientemente ELECTRA [15].

La razón por la que los modelos pre-entrenados funcionan bien es por su exposición a masivas cantidades de datos en la tarea original. El método por el cual se transfiere el modelo entrenado en una tarea hacia otras se conoce como Aprendizaje por Transferencia.

- **Aprendizaje por Transferencia:** Es una técnica de aprendizaje en la que un modelo que ha sido entrenado para una tarea, es tomado y es entrenado para realizar una tarea diferente. Es el siguiente paso a utilizar cuando se tiene un modelo pre-Entrenado, como el de BERT o ELECTRA, los cuales se encargan del modelamiento del lenguaje. Sobre estos modelos pre-Entrenados se realiza un procesamiento de aprendizaje por transferencia para las distintas tareas de NLP que pueden ser: Preguntas y Respuestas, Reconocimiento de Entidades, Analisis de Sentimientos.

2.3.1. BERT

BERT (Bidirectional Encoder Representations from Transformers) es un modelo del 2018, desarrollado por los investigadores de Google. Se diferencia de los anteriores modelos en que las secuencias de entrada pueden ser procesadas de manera bidireccional, utilizando dos codificadores sobre una entrada. Mediante esta técnica se reduce la probabilidad de cometer un error antes de una predicción [2]. BERT está preentrenado usando dos tareas no supervisadas que solo requieren de un corpus de texto y no de etiquetas hechas a mano. Estas tareas son:

- **Modelado de lenguaje enmascarado:** Consiste en reemplazar aleatoriamente palabras de la entrada por el símbolo [MASK] y luego predecir cual ha sido la palabra original basada en el contexto de las palabras restantes, por ejemplo en la oración : *Carlos es un [MASK] de la escuela*. El modelo nos retornará como una predicción de alta probabilidad a las palabras *alumno* o *maestro*.
- **Predicción de la siguiente oración:** Para capturar las relaciones a nivel de oración, se agrega la tarea no supervisada de la predicción de la siguiente oración. Para generar ejemplos de esta tarea en cada ejemplo el 50 % de las veces la siguiente oración es la siguiente oración real y la otra mitad del tiempo es una oración aleatoria del corpus.

Para hacer uso de BERT, 2 etapas deben ser seguidas:

- **Pre-entrenamiento:** Es la etapa en la que el modelo es entrenado en data sin etiquetas, este entrenamiento se desarrolla sobre las tareas de Modelamiento de Lenguaje Enmascarado y la de Predicción de la Siguiente Oración.
- **Fine-tuning:** El modelo BERT es inicializado con los parámetros pre-entrenados seguidos por una técnica de transferencia de aprendizaje usando la data de las nuevas tareas que pueden ser: Clasificación de textos, Preguntas y respuestas, etc.

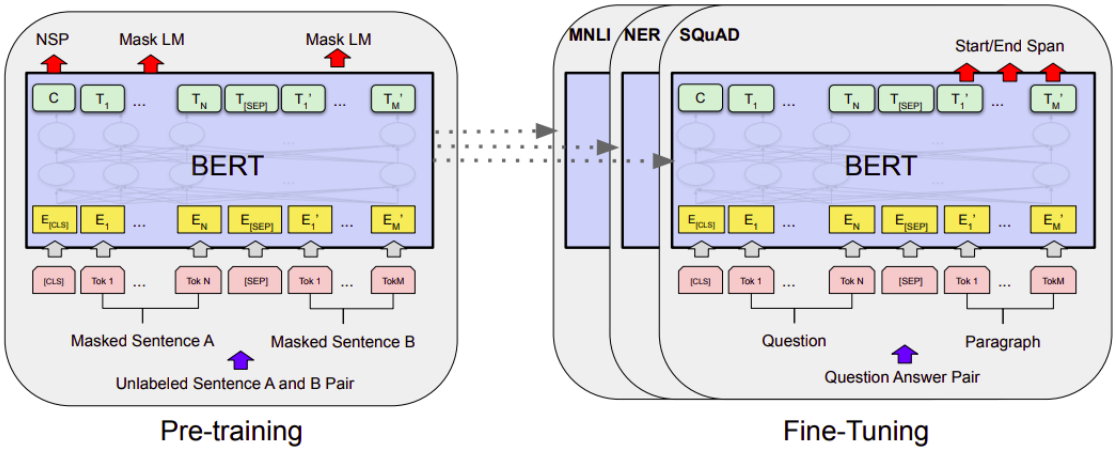

Figura 2.9: Procesos de entrenamiento sobre BERT [14]

Por su buen desempeño en varias tareas de NLP, se han desarrollado variantes de BERT como son RoBERTa [16] y ALBERT [17] las cuales ofrecen mejores resultados pero con un mayor coste computacional por el incremento en el número de párametros y tiempo de entrenamiento.

2.3.2. ELECTRA

ELECTRA (Efficiently Learning an Encoder that Classifies Token Replacements Accurately) [15] es un modelo del 2020 que contiene un método de pre-entrenamiento superior a modelos anteriores como GPT, BERT, RoBERTa, ALBERT. Además, ofrece un mejor desempeño en distintas tareas de NLP con menor tamaño del modelo y menor necesidad de poder de computo.

ELECTRA reemplaza el modelo de lenguaje enmascarado de BERT por la técnica de Detección de Token Reemplazado, el cual por experimentos demuestra ser más eficiente y producir mejores resultados.
El pre-entrenamiento de ELECTRA consiste en reemplazar aleatoriamente algunos tokens de la entrada con alternativas aceptables conseguidas de una pequeña red generadora que utilice el modelado de lenguaje enmascarado. Después, un modelo discriminador es entrenado para predecir si cada token de la entrada modificada ha sido reemplazado por la red generadora o no. Este nuevo método de pre-entrenamiento es más eficiente que el de BERT, porque aprende de todos los tokens de entrada a diferencia de BERT que enmascara solo el 15 % de tokens.

- **Pre-entrenamiento: Replaced Token Detection:** Este método entrena 2 redes neuronales, un Generador y un Discriminador. El generador es entrenado para realizar un modelo de lenguaje enmascarado, seleccionando aleatoriamente un conjunto de elementos de la entrada y reemplazándolos por el token [MASK]. Luego, el generador aprende a maximizar la probabilidad de los tokens enmascarados. Finalmente, el discriminador es entrenado para distinguir los tokens que han sido reemplazados por el generador.

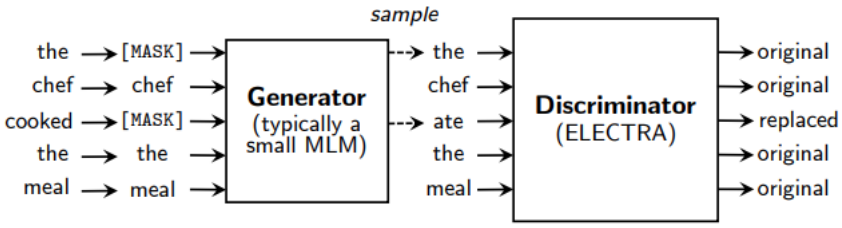

Figura 2.10: Arquitectura Generador Discriminador usado en ELECTRA [15]

2.4. Conjunto de Datos

El conjunto de datos a utilizar es el resultado de un paper que aplica una técnica de traducción automática de inglés a español al Conjunto de Datos de Preguntas y Respuestas SQuAD de la universidad Stanford [18], el cual está compuesto por preguntas anotadas de una selección de artículos de Wikipedia. La respuesta a cada pregunta es una sección del texto (un intervalo) de la lectura correspondiente. El conjunto de datos contiene alrededor de 100 mil pares de preguntas y respuestas de 442 temas. Para cada tema hay un conjunto de contextos y cada contexto tiene pares de preguntas y respuestas, los cuales se anotan marcando el tramo o la parte de texto que responde a la pregunta. Los archivos pueden ser descargados en formato .json desde la página de github del autor del paper [19].

	Español
Contexto	Fryderyk Chopin nació en **Zelazowa Wola**, 46 kilómetros al oeste de Varsovia, en lo que entonces era el Ducado de Varsovia, un estado polaco establecido por Napoleón. El registro de bautismo de la parroquia da su cumpleaños el **22 de febrero de 1810**, y cita sus nombres en latín **Fridericus Franciscus** (en polaco, Fryderyk Franciszek). Sin embargo, el compositor y su familia utilizaron la fecha de nacimiento 1 de marzo, [n 2] que ahora se acepta generalmente como la fecha correcta.
Pregunta	1) ¿En qué pueblo nació Frédéric?
Respuesta	1) **Zelazowa Wola,**
Pregunta	2) ¿Cuándo se registró su cumpleaños
Respuesta	2) **22 de febrero de 1810,**
Pregunta	3) ¿Cuál es la forma latina del nombre de Chopin?
Respuesta	3) **Fridericus Franciscus**
Contexto	Durante el período de las Cinco Dinastias y los Diez Reinos de China **(907-960)**, mientras que el reino político fracturado de China no vio ninguna amenaza en un Tíbet que estaba en el mismo desorden político. Pocos documentos que involucran contactos sino-tibetanos sobreviven de la dinastía Song (960-1279). Los Song estaban mucho más preocupados por contrarrestar los estados enemigos del norte de **la dinastía Liao gobernada por Kitán** (907-1125) y la dinastía Jin **gobernada por Jurchen** (1115-1234).
Pregunta	1) ¿Cuándo tuvo lugar el período de las Cinco Dinastias y los Diez Reinos de China?
Respuesta	1) **(907-960),**
Pregunta	2) ¿Quién gobernó la dinastía Liao
Respuesta	2) **la dinastía Liao gobernada por Kitán**
Pregunta	3) ¿Quién gobernó la dinastía Jin?
Respuesta	3) **gobernada por Jurchen**

Cuadro 2.1: Ejemplos de contexto, preguntas y respuestas seleccionados de la data de entrenamiento de SQuAD en español

Revisión de la Literatura

En el presente capítulo se desarrollan los criterios de búsqueda utilizados para identificar el estado del arte, y las arquitecturas alternativas desarrolladas con respecto a la tarea de Preguntas y Respuestas.

3.1. Objetivo de la revisión

El objetivo de la revisión es identificar y seleccionar los libros y papers desarrollados para resolver la problemática planteada.

3.1.1. Cadenas de Búsqueda

La búsqueda se realizó en las bases de datos de Scopus, ACL, Arxiv y ACM.

- TITLE-ABS-KEY ((question answering OR QA) AND BERT)
- TITLE-ABS-KEY ((question answering OR QA) AND ELECTRA)
- TITLE-ABS-KEY ((question answering OR QA) AND LSTM)
- TITLE-ABS-KEY ((question answering OR QA) AND Attention Mechanism)
- TITLE-ABS-KEY (NLP AND libraries)

3.1.2. Preguntas de Revisión

- ¿Cuáles son las arquitecturas base de redes neuronales para las tareas de NLP de preguntas y respuestas?
- ¿Cómo se pueden vectorizar las palabras para ingresar a las arquitecturas de redes neuronales?
- ¿Cuáles son los modelos que ofrecen el estado del arte para la tarea de NLP de preguntas y respuestas?
- ¿Cuáles son las métricas para medir un buen desempeño de los modelos en la tarea de preguntas y respuestas?
- ¿Cuáles son las líbrerías más útiles para realizar investigaciones de NLP?

3.2. Resultados de la revisión

Se obtuvieron 12 artículos de los cuales 5 se consideraron los más relevantes para responder las preguntas de revisión.

En [20] se hace mención de DrQA de Stanford, el cual fue una de las primeras arquitecturas de tipo LSTM entrenadas para la tarea de preguntas y respuestas, su implementación se desarrolló con 3 capas bidireccionales LSTM, un vector de capas ocultas de dimensión 128, tanto para el contexto como la pregunta. Se utilizó un tamaño de batch de 32, una función de optimización Adamax y un dropout de 0.3 aplicado a las palabras vectorizadas (word embeddings) y las unidades ocultas de las celdas LSTM.

También se hace mención que el método para vectorizar las palabras entrantes a la arquitectura de tipo LSTM, es el conocido como Word Embeddings. Además en el paper se menciona que se usó en específico Glove, que ha sido entrenado para 840 billones de palabras obtenidas de la web. Además se indica que para la tarea estudiada para este proyecto, se aplica el mecanismo de atención cruzada entre las preguntas y respuestas resultantes de la vectorización por Word Embeddings.

En [21] se evaluan distintos modelos pre-entrenados para el idioma árabe y la tarea de Análisis de Sentimientos y Detección de Sarcasmo. Se muestra que los modelos pre-entrenados que ofrecen el estado del arte para diversas tareas de NLP son BERT, GPT y ELECTRA. Siendo ELECTRA el que ofrece los mejores resultados, además de ser más eficiente en su costo computacional. Además se hace mención de los mejores resultados obtenidos en el uso de modelos de deep learning con Word Embeddings frente a modelos clásicos de machine learning como SVMs; y que los modelos basados en Transformers muestran mejores resultados que los basados en deep learning.

También se hace mención que el modelo basado en BERT: AraBERT supera a los otros modelos basados en Transformers. Con respecto al entrenamiento con la técnica de Transferencia de Aprendizaje si indica que ELECTRA es uno de los modelos más ligeros y de rápido entrenamiento, llegando a obtener un desempeño similar a modelos mucho más grandes como GPT. Se menciona del rol que tiene el tamaño de los modelos mostrando que mientrás más pesado es el modelo, mayor es el consumo de memoria y más lento su entrenamiento.

En [22] se hace la comparación de BERT y ELECTRA small, que es una versión de ELECTRA reducido debido al menor número de parámetros de entrenamiento, llegando a ser 8 veces más pequeño que el modelo BERT y 8 veces más rápido en el tiempo de inferencia. Además se hace mención del cuello de botella que significa tener modelos grandes para aplicaciones de tiempo real como las tareas de preguntas y respuestas. Por ese motivo proponen el entrenamiento de modelos más ligeros como ELECTRA small, donde logran alcanzar resultados similares a los modelos BERT para la tarea de minería de textos de conjuntos de datos biomédicos.

En [18] se hace mención de que las métricas para medir la precisión de los modelos de sistemas de preguntas y respuestas son exact match y F1. Exact match (EM) se encarga de medir el porcentaje de predicciones que coinciden exactamente con las respuestas correctas. F1 se encarga de medir el promedio del solapamiento entre la predicción y la respuesta correcta. También se hace mención que para la tarea de preguntas y respuestas el desempeño de un adulto humano promedio es 77 % para exact match y 86.8 % para F1. A su vez se indica que el modelo más sencillo que una regresión logística obtiene una métrica de 51 % para F1.

En [23] se hace mención de la librería de código abierto Transformers, cuya finalidad es facilitar la investigación de estas recientes tecnologías a la comunidad internacional. La librería consiste en una colección de modelos pre-entrenados, conjuntos de datos y también posee APIs para aplicaciones en la industria. La librería esta disponible en:

https://github.com/huggingface/transformers.

También se hace mención de cómo la arquitectura de Transformers ha generado una revolución en el área de investigación de NLP, superando los modelos basados en redes neuronales y redes convolucionales. Siendo las tareas donde se obtuvieron mayores ganancias en exactitud: Clasificación de Texto, Traducción de máquina, sumarización entre otras.

Cada modelo de la librería esta compuesto por 3 bloques principales: un tokenizador, que convierte las palabras a índices; un transformer, que convierte los índices en vectores contextualizados, es decir que una palabra puede tener un representación vectorial distinta dependiendo de las palabras que la rodeen; una cabeza, que usa los vectores contextualizados para realizar una tarea específica de predicción, esta cabeza puede variar dependiendo de la tarea para la cual se entrene el modelo base mediante la técnica de Transferencia de Aprendizaje.

Entre los principales modelos soportados por esta librería para desarrollar experimentos tenemos: BERT, RoBERTa GPT, GPT2, XLNet, BART, T5, MarianMT y ELECTRA. Otra caracteristíca es la librería es que el desarrollo de proyectos es más amigable con el framework de Pytorch; sin embargo, también tiene soporte para usar modelos con TensorFlow.

Experimentación y Resultados

En el presente capítulo se presentan los resultados luego de cumplir con los objetivos propuestos.

4.1. Arquitectura de Red Neuronal LSTM con mecanismos de Atención: DrQA

4.1.1. Introducción

En esta sección se desarrollará el resultado esperado correspondiente al objetivo O1. En este se realiza el entrenamiento de la arquitectura de la red neuronal de tipo LSTM con mecanismos de atención.

4.1.2. DrQA

Para la vectorización de las palabras se utilizó el archivo de Word Embeddings de palabras en español proporcionado por Kaggle, el cual esta entrenado sobre 1 millón de palabras y donde cada palabra esta representada por un vector de números flotantes de dimensión 300.

Los hiperparámetros usados para el entrenamiento fueron los siguientes:

Hiperparámetro	Valor
Número de épocas	10
Optimizador	Adam
Tamaño de Batch	128
Dimensión de capa oculta	128
Número de direcciones	2
Dropout	3
Dimension de word embeddings	300

Las métricas obtenidas fueron:

Métrica	Resultado
EM	41.7
F1	59.5

Se observa una mejora en la métrica F1 de 8.5 % con respecto a un simple modelo de regresión lineal como se vió en la revisión de literatura. Sin embargo, estas métricas nos indican los resultados mínimos esperados para los modelos pre-entrenados.

4.2. Modelos Pre-entrenados de tipo Transformers

4.2.1. Introducción

En esta sección se desarrollará el resultado esperado correspondiente al objetivo O2. En este se realiza el entrenamiento con la técnica de Transferencia de Aprendizaje para los modelos pre-entrenados de BERT, RoBERTa y ELECTRA en español utilizando la librería de Huggingface.

4.2.2. Modelo BERT en español

Se desarrollo la técnica de Transferencia de Aprendizaje sobre el modelo entrenado para español BETO, para la tarea de preguntas y respuestas, con el dataset SQUAD en español squad_es.

Los hiperparámetros usados para el entrenamiento fueron los siguientes:

Hiperparámetro	Valor
Número de épocas	2
Optimizador	AdamW
Tamaño de Batch	8
Taza de Aprendizaje	3e-5

Las métricas obtenidas fueron:

Métrica	Resultado
EM	59.5
F1	76.6

Se observa una mejora de 17.1 % en la métrica F1 y de 17.8 % en la métrica exact match con respecto a la arquitectura DrQA.

4.2.3. Modelo RoBERTa en español

Se desarrollo la técnica de Transferencia de Aprendizaje sobre el modelo entrenado para español Ruperta, para la tarea de preguntas y respuestas, con el dataset SQUAD en español squad_es.

Los hiperparámetros usados para el entrenamiento fueron los siguientes:

Hiperparámetro	Valor
Número de épocas	2
Optimizador	AdamW
Tamaño de Batch	8
Taza de Aprendizaje	3e-5

Las métricas obtenidas fueron:

Se observa una mejora de 6.9 % en la métrica F1 y de 8.1 % en la métrica exact match con respecto a la arquitectura DrQA. Sin embargo, sus métricas son menores a la de BERT es español, lo cual

Métrica	Resultado
EM	49.8
F1	66.4

significaría que el modelo en español de RoBERTa no ha sido entrenado con la cantidad adecuada de texto como en el modelo en inglés.

4.2.4. Modelo ELECTRA en español

Se desarrollo la técnica de Transferencia de Aprendizaje sobre el modelo entrenado para español Electricidad, para la tarea de preguntas y respuestas, con el dataset SQUAD en español squad_es.

Los hiperparámetros usados para el entrenamiento fueron los siguientes:

Hiperparámetro	Valor
Número de épocas	2
Optimizador	AdamW
Tamaño de Batch	8
Taza de Aprendizaje	3e-5

Las métricas obtenidas fueron:

Métrica	Resultado
EM	58.4
F1	75.1

Se observa una mejora de 15.6 % en la métrica F1 y de 16.4 % en la métrica exact match con respecto a la arquitectura DrQA.

Adicionalmente se realizó el entrenamiento para una versión reducida de ELECTRA: ELECTRA small, que cuenta con menos parámetros de entrenamiento pero unas métricas aceptables para su tamaño, aproximadamente 10 veces menor que el modelo base.

Las métricas obtenidas fueron:

Métrica	Resultado
EM	46.1
F1	64.2

Se observa una mejora de 4.7 % en la métrica F1 y de 4.4 % en la métrica exact match con respecto a la arquitectura DrQA. Esto nos muestra que los modelos pre- entrenados, incluyendo los de pequeño tamaño ofrecen mejores métricas que las arquitecuras de redes neuronales de tipo LSTM.

4.3. Resultados

4.3.1. Introducción

En esta sección se desarrollará el resultado esperado correspondiente al objetivo O3. En este se realiza la comparativa de la arquitectura de redes neuronales LSTM: DrQA y los modelos BERT, RoBERTa y ELECTRA en español entrenados para la tarea de preguntas y respuestas.

4.3.2. Comparativa de Resultados

En las siguientes figuras se muestran los resultados del entrenamiento de los distintos modelos estudiados: en color fucsia ELECTRA base, en color celeste BERT, en color azul RoBERTa y en color naranja ELECTRA small.

En el experimentó se realizó un entrenamiento de 6 épocas sobre ELECTRA base. Como resultado del experimento se observa que desde la época 3 ocurre el fenómeno de sobre ajuste. El mismo fenómeno se observa en los modelos de BERT y RoBERTa. Este fenómeno ocurre cuando el modelo aprende a predecir mejor los datos de entrenamiento, pero pierde una predicción para datos no observados. Las mejores métricas se obtienen al finalizar el entrenamiento de los modelos en la segunda época.

En las curvas de entrenamiento se observa que BERT tiene un mejor aprendizaje que el resto de modelos, siguiendolé muy de cerca ELECTRA-base.

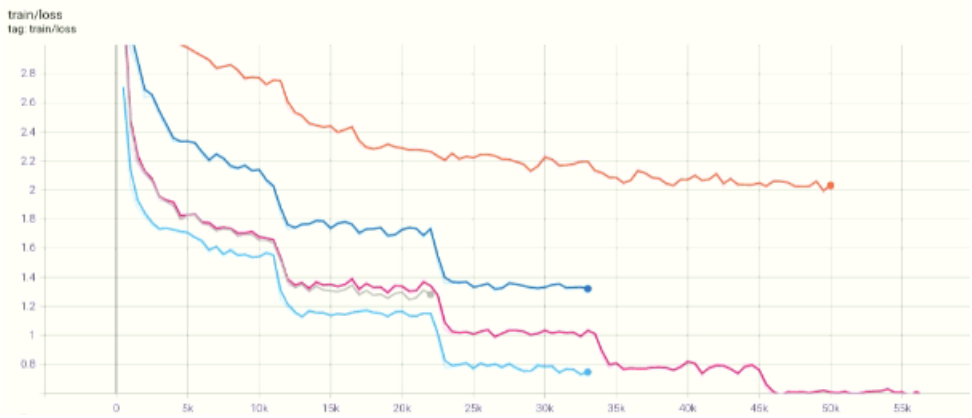

Figura 4.1: Comparativa de los modelos de las funciones de pérdida en la etapa de entrenamiento vs número de pasos

En la tabla de comparativa podemos observar que DrQA posee un tiempo de inferencia que es 10 veces más pequeño así como un peso de su arquitectura 5 veces más pequeña que los modelos BERT, RoBERTa y ELECTRA base. Sin embargo, un problema de su utilización en un sistema de producción es el tama´ no del archivo de vectorización de palabras (word embeddings) que es del orden de 3 Gb, lo cual genera un alto consumo de recursos.

Se observa que el estado del arte para la tarea de NLP de Preguntas y Respuestas en español esta dada por BERT que posee un tamaño del modelo y un tiempo de inferencia muy similar al modelo de ELECTRA, el cual es el segundo modelo con mejores métricas.

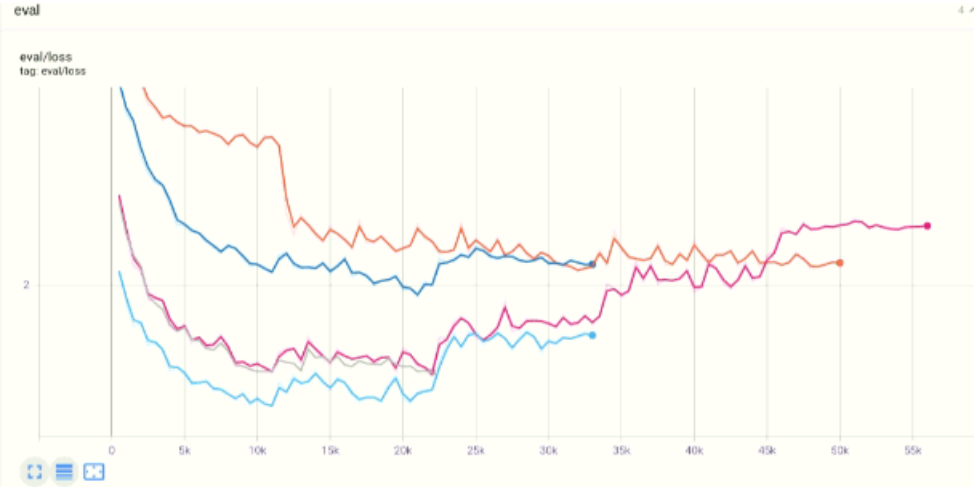

Figura 4.2: Comparativa de los modelos de las funciones de pérdida en la etapa de validación vs número de pasos

Se observa que el modelo ELECTRA-small produce métricas que superan al modelo de DrQA y a su vez posee un menor peso, lo cuál lo haría útil para casos de preguntas y respuestas sencillas.

De la tabla comparativa, además podemos añadir que:

- El menor tiempo de entrenamiento y tiempo de inferencia del modelo DrQA se debe a la menor cantidad de parámetros de entrenamiento con respecto a los modelos pre-entrenados.
- Las métricas obtenidas para ELECTRA y BERT son muy cercanas, esto se debe a la eficiente arquitectura de ELECTRA. Sin embargo, la curva de apredizaje de BERT muestra que el modelo proporcionado por HuggingFace tiene ha sido entrenado con mayor data que el modelo de ELECTRA, por ese motivo BERT muestra mejores resultados para el idioma español.
- El tiempo de entrenamiento de los modelos BERT y ELECTRA muestran similares características debido al similar tamaño del modelo, lo que nos genera tiempos de inferencia similares.
- Para un sistema de preguntas y respuestas en español los mejores modelos a usar son ELECTRA-base y BERT en español.

Modelo	EM	F1	Peso	T.Entren. (1 época)	T. de Inferencia
DrQA	41.7	59.5	128 Mb	10 min	0.002 s.
BERT en español	59.5	76.6	438 Mb	4h	0.2 s.
RoBERTa en español	49.8	66.4	505 Mb	5h	0.2 s.
ELECTRA-small en español	46.1	64.2	53 Mb	1h	0.04 s.
ELECTRA-base en español	58.4	75.1	437 Mb	4h	0.21 s.

4.4. Prototipo de Despliegue

4.4.1. Introducción

En esta sección se desarrollará el resultado esperado correspondiente al objetivo O4. En este se realiza un prototipo de un sistema de preguntas y respuestas mediante el uso de la librería de código abierto de Python Streamlit https://streamlit.io/.

4.4.2. Prototipo

En las siguientes figuras se muestra el prototipo desarrollado, donde se cargan los textos a analizar que son el contexto y una pregunta. Posteriormente se escoge el modelo que ha sido pre-entrenado para realizar la inferencia y finalmente se muestra la salida del modelo.

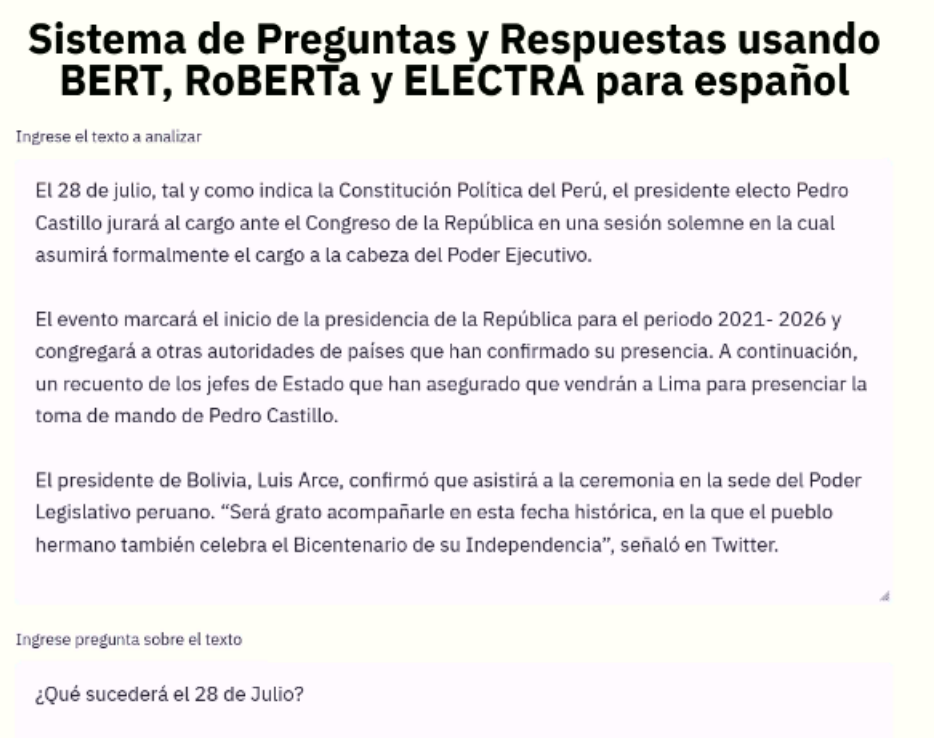

Figura 4.3: Prototipo de Sistema de Preguntas y Respuestas: Contexto y Pregunta

En distintos ejemplos analizados se puede observar que ELECTRA base y BERT para español ofrecen las mejores respuestas con un tiempo de inferencia similar.

Figura 4.4: Inferencia del Modelo BERT

Figura 4.5: Inferencia del Modelo RoBERTa

Figura 4.6: Inferencia del Modelo ELECTRA base

Figura 4.7: Inferencia del Modelo ELECTRA small

Conclusiones y Trabajos Futuros

5.1. Conclusiones

- Se experimentó con distintas arquitecturas de redes neuronales, entre ellas: redes recurrentes LSTM con mecanismos de atención como DrQA y arquitecturas basadas en Transformers aplicados modelos pre-entrenados como BERT, RoBERTa y ELECTRA en el lenguaje español.
- Se aplicó la técnica de trasferencia de aprendizaje para la tarea de Preguntas y Respuestas con el conjunto de datos SQUAD v1, traducido al español.
- Se desarrolló una comparativa de las métricas de las arquitecturas de redes neuronales estudiadas, donde se observó que ELECTRA en español y BERT en español ofrecen los mejores desempe´ nos debido a sus métricas y su tiempo de inferencia.
- Se desarrolló un prototipo de despliegue de un sistema de preguntas y respuestas incluyendo a los modelos que nos brindan las mejores métricas.

5.2. Trabajos Futuros

- Mejorar el estudio agregando otros conjuntos de datos como SQUAD v2 [24], MLQA [25], XQUAD [26], etc.
- Con los modelos que mostraron mejores métricas, realizar un sistema de preguntas y respuestas de dominio abierto en el cual el contexto para una pregunta será un documento largo.
- En el campo de aplicación de la industria, se puede realizar un sistema de chatbot que nos permita realizar consultas y provea respuestas de manera automática en base a un dominio específico.

Bibliografía

[1] Chethan Kumar GN. *NLP vs NLU vs NLG (Know what you are trying to achieve) NLP engine (Part-1)*. Jun. de 2019. URL: https://towardsdatascience.com/nlp-vs-nlu-vs-nlg-know-what-you-are-trying-to-achieve-nlp-engine-part-1-1487a2c8b696.

[2] Navin Sabharwal y Amit Agrawal. *Hands-on question answering systems with BERT: applications in neural networks and natural language processing*. Apress, 2021.

[3] *UNIDAD 4 REDES NEURONALES - INTELIGENCIA ARTIFICIAL*. URL: https://sites.google.com/site/mayinteligenciartificial/unidad-4-redes-neuronales.

[4] Aravind PaiAravind is a sports fanatic. His passion lies in developing data-driven products for the sports domain. He strongly believes that analytics in sports can be a game-changer. *ANN vs CNN vs RNN: Types of Neural Networks*. Oct. de 2020. URL: https://www.analyticsvidhya.com/blog/2020/02/cnn-vs-rnn-vs-mlp-analyzing-3-types-of-neural-networks-in-deep-learning/.

[5] Chenguang Zhu. *Machine reading comprehension: algorithms and practice*. Elsevier, 2021.

[6] Ralf C. Staudemeyer y Eric Rothstein Morris. "Understanding LSTM - a tutorial into Long Short-Term Memory Recurrent Neural Networks". En: *CoRR* abs/1909.09586 (2019). arXiv: 1909.09586. URL: http://arxiv.org/abs/1909.09586.

[7] Gabriel Loye. *Long Short-Term Memory: From Zero to Hero with PyTorch*. Ago. de 2019. URL: https://blog.floydhub.com/long-short-term-memory-from-zero-to-hero-with-pytorch/.

[8] Simeon Kostadinov. *Understanding Encoder-Decoder Sequence to Sequence Model*. Nov. de 2019. URL: https://towardsdatascience.com/understanding-encoder-decoder-sequence-to-sequence-model-679e04af4346.

[9] Lilian Weng. "Attention? Attention!" En: *lilianweng.github.io/lil-log* (2018). URL: http://lilianweng.github.io/lil-log/2018/06/24/attention-attention.html.

[10] Ashish Vaswani y col. *Attention Is All You Need*. 2017. arXiv: 1706.03762 [cs.CL].

[11] Bryan McCann y col. *Learned in Translation: Contextualized Word Vectors*. 2018. arXiv: 1708.00107 [cs.CL].

[12] Matthew E. Peters y col. *Deep contextualized word representations*. 2018. arXiv: 1802.05365 [cs.CL].

[13] Tom B. Brown y col. *Language Models are Few-Shot Learners*. 2020. arXiv: 2005.14165 [cs.CL].

[14] Jacob Devlin y col. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. 2019. arXiv: 1810.04805 [cs.CL].

[15] Kevin Clark y col. *ELECTRA: Pre-training Text Encoders as Discriminators Rather Than Generators*. 2020. arXiv: 2003.10555 [cs.CL].

[16] Yinhan Liu y col. *RoBERTa: A Robustly Optimized BERT Pretraining Approach*. 2019. arXiv: 1907.11692 [cs.CL].

[17] Zhenzhong Lan y col. *ALBERT: A Lite BERT for Self-supervised Learning of Language Representations*. 2020. arXiv: 1909.11942 [cs.CL].

[18] Pranav Rajpurkar y col. *SQuAD: 100,000+ Questions for Machine Comprehension of Text*. 2016. arXiv: 1606.05250 [cs.CL].

[19] Casimiro Pio Carrino, Marta R. Costa-jussa y Jose A. R. Fonollosa. *Automatic Spanish Translation of the SQuAD Dataset for Multilingual Question Answering*. 2019. arXiv: 1912.05200 [cs.CL].

[20] Danqi Chen y col. *Reading Wikipedia to Answer Open-Domain Questions*. 2017. arXiv: 1704.00051 [cs.CL].

[21] Ibrahim Abu Farha y Walid Magdy. "Benchmarking Transformer-based Language Models for Arabic Sentiment and Sarcasm Detection". En: *Proceedings of the Sixth Arabic Natural Language Processing Workshop*. Kyiv, Ukraine (Virtual): Association for Computational Linguistics, abr. de 2021, págs. 21-31. URL: https://aclanthology.org/2021.wanlp-1.3.

[22] Ibrahim Ozyurt. "On the effectiveness of small, discriminatively pre-trained language representation models for biomedical text mining". En: (mayo de 2020). DOI: 10.1101/2020.05.20.107003.

[23] Thomas Wolf y col. *HuggingFace's Transformers: State-of-the-art Natural Language Processing*. 2020. arXiv: 1910.03771 [cs.CL].

[24] Pranav Rajpurkar, Robin Jia y Percy Liang. *Know What You Don't Know: Unanswerable Questions for SQuAD*. 2018. arXiv: 1806.03822 [cs.CL].

[25] Patrick Lewis y col. *MLQA: Evaluating Cross-lingual Extractive Question Answering*. 2020. arXiv: 1910.07475 [cs.CL].

[26] Mikel Artetxe, Sebastian Ruder y Dani Yogatama. *On the Cross-lingual Transferability of Monolingual Representations*. 2020. arXiv: 1910.11856 [cs.CL].