Capítulo 6

Parte 3: Chatbots conversacionales inteligentes de texto

Conceptos básicos

- Los **modelos de texto** son modelos basados en el PLN. Son capaces de realizar tareas relacionadas con la lectura y escritura, como hacer resúmenes, mapas mentales, traducciones, etc.
  - Por ejemplo, los modelos GPT.
- Los modelos de texto **se entrenan** para predecir la palabra siguiente en una secuencia de texto dada previamente. Esto da como resultado lo que se conoce como **modelo de lenguaje.**
- El modelado de lenguaje se basa en que las palabras en un idioma no se usan de forma aleatoria, sino que las palabras están **relacionadas entre sí de una manera predecible.**
  - Es decir, en un determinado contexto, después de una determinada palabra es más probable que venga una palabra que otra.

Conceptos básicos

- Los **grandes modelos de lenguaje** (LLM) se entrenan con enormes conjuntos de datos de texto y usan redes neuronales profundas
  - para procesar la información y aprender patrones en el lenguaje.
  - Por ejemplo, los modelos GPT y BERT son LLM.
  - Producir LLM consume muchos recursos.
  - Los GPT usan redes neuronales tipo transformer para el modelado de lenguaje en tareas específicas de PLN.
    - Los GPT son capaces de generar texto coherente de alta calidad, así como realizar una amplia variedad de tareas de PLN como resúmenes, mapas mentales, traducciones, clasificaciones de texto, respuestas a preguntas, etc.

Conceptos básicos

- Los **modelos conversacionales** o **de dialogo** son modelos de generación de texto que están preparados para poder conversar.
  - P.ej: ChatGPT es un modelo de diálogo o conversacional, que deriva de un modelo de texto.
- Los **chatbots de IA generativa** utilizan grandes modelos lingüísticos (LLM) para generar respuestas basadas en sus entradas.
  - Se entrenan con conjuntos de datos masivos que contienen miles de millones de frases y oraciones.
  - Los LLM aprovechan el aprendizaje profundo (basado en redes neuronales) y el PLN.
  - Esto ayuda al chatbot a entender y producir respuestas similares a las humanas.

Etapas para crear un modelo de IA

- **Etapa 1: preparar los datos.**
  - Los datos que se usarán serán muchos datos, potencialmente petabytes de datos en docenas de dominios. Los datos pueden combinar datos de código abierto y datos propios.
    - Suelen considerarse grandes cantidades de texto de diferentes fuentes como libros, artículos, sitios web, etc.
    - P.ej: Wikipedia, libros electrónicos, noticias.
  - La etapa1 implica tareas de procesamiento de datos:
    1. Se describen los datos (qué son);
    2. Se filtran (por ejemplo, contenido abusivo, material protegido por derechos de autor), ya que no queremos entrenar el modelo con ellos.
    3. Se eliminan los datos duplicados.
  - Como resultado de la etapa 1 obtenemos una **pila de datos base**.
  - La pila de datos base no está etiquetada y entrenamos el modelo de IA con ella.

Etapas para crear un modelo de IA

- **Etapa 2: entrenar el modelo.**
  1. Elegimos una **arquitectura del modelo** como Transformers.
     - Elija el modelo que se ajuste a su caso de uso.
  2. Se **empareja** la pila de datos con ese modelo.
  3. Tokenizamos la pila de datos.
     - Los modelos de lenguaje trabajan con tokens y no con palabras.
     - Una pila de datos puede dar lugar a billones de tokens.
  4. Se inicia el **proceso de entrenamiento** sobre esos tokens.
     - Este proceso puede llevar mucho tiempo, dependiendo del tamaño del modelo.
     - UN LLM a gran escala puede llevar meses con muchos miles de GPU.
     - Una vez hecho, el mayor tiempo y el mayor coste computacional han quedado atrás.

Etapas para crear un modelo de IA

- **Etapa 2: entrenar el modelo (cont)**

5. Durante el entrenamiento, el modelo aprende cómo los tokens **se relacionan entre sí**.

   - Esto incluye la identificación de estructuras gramaticales, contextos semánticos y patrones de uso del lenguaje.
   - El modelo no solo memoriza ejemplos específicos, sino que también **generaliza** a partir de ellos, permitiendo generar respuestas coherentes y contextualmente relevantes a nuevas preguntas o prompts.
   - El entrenamiento implica varias **fases**, como tokenización, codificación posicional, cálculo de autoatención y generación de representaciones contextuales.
   - El modelo **ajusta sus parámetros a través de un proceso iterativo** para mejorar su capacidad de generar texto.

Etapas para crear un modelo de IA

- **Fase 3: Validación.**
  - **Evaluar el modelo** entrenado implica **ejecutar el modelo y comprobar su rendimiento** con respecto a un conjunto de puntos de referencia (benchmarks) que ayudan a determinar la **calidad del modelo**.
    - Entonces podemos crear una tarjeta de modelo que diga: este es el modelo que se ha entrenado y estos son los puntos de referencia que ha alcanzado.
  - Para validar grandes modelos de lenguaje, se utilizan varias **métricas especializadas** que evalúan distintos aspectos del rendimiento del modelo. Por ejemplo, hay métricas para:
    - Medir la capacidad del modelo para **predecir una muestra de texto** (o sea, que también predice el siguiente token en una secuencia).
    - Medir la **similitud** entre el texto generado por el modelo y un texto de referencia.
    - Medir el porcentaje de **predicciones que coinciden** con las respuestas de referencia. Es común para tareas de respuestas a preguntas.
    - Medir la calidad de las traducciones automáticas.

Etapas para crear un modelo de IA

- **Etapa 4: Puesta a punto.**
  - El desarrollador de aplicaciones se involucra con el modelo generado.
    - Puede dar instrucciones que provoquen un buen rendimiento del modelo.
    - Pueden proporcionar datos locales adicionales para ajustar el modelo y mejorar su rendimiento.
  - Esta etapa puede ser mucho más rápida que crear un modelo desde cero.
  - El **ajuste fino** de un modelo pre-entrenado tiene el propósito de adaptarlo a una tarea o dominio específico.
    - Se usa un conjunto de datos específico para ajustar el modelo pre-entrenado y mejorar su rendimiento en una tarea particular.

Etapas para crear un modelo de IA

- **Etapa 4: Puesta a punto (cont)**
  - La **adaptación a nuevos** datos tiene el propósito de mantener el modelo actualizado con información reciente y relevante.
    - Se reentrena o ajusta el modelo existente con nuevos datos que no estaban disponibles durante el entrenamiento inicial.
  - La puesta a punto puede involucrar tanto el ajuste fino como la adaptación a nuevos datos en diferentes momentos:
    - Inicialmente usa puesta a punto y luego continuamente implementas la adaptación a nuevos datos para mantener el modelo actualizado y relevante con el tiempo.

Etapas para crear un modelo de IA

- **Etapa 5: Despliegue del modelo:**
  - El modelo puede desplegarse en una **nube pública** o integrarse en una **aplicación o servicio** existente.
  - Se puede seguir mejorando el modelo con el tiempo con **actualizaciones periódicas** para ajustar el modelo con nuevos datos y mejorar su rendimiento.
  - Aquí hay que preocuparse con el **balance de carga**, o sea, asegurarse que el sistema pueda manejar múltiples solicitudes simultáneas.
  - Además, hay que preocuparse con **optimizar los recursos de computación** para que la respuesta a solicitudes sea eficiente.

Arquitectura de los chatbot generativos inteligentes

- **Componentes de un chatbot conversacional de texto:**
  - **Interfaz del usuario**: permite a los usuarios escribir preguntas o solicitudes en un cuadro de texto, donde el bot responde casi instantáneamente.
    - Esta facilidad de uso es fundamental para la experiencia del usuario.
  - **Motor de PLN :**
    - **Hace preprocesamiento de texto**: divide texto en tokens, remueve stop words, maneja puntuación, y prepara el input para el modelo.
    - **Reconoce la intención del usuario** detrás del prompt (p.ej: hacer pregunta, pedir una acción, etc.),
    - **Hace reconocimiento de entidades**: extrae las entidades claves del texto.

Arquitectura de los chatbot generativos inteligentes

- **Componentes de un chatbot conversacional de texto (cont.):**
  - **Modelo grande de lenguaje**: El LLM genera texto (crea texto basado en input, produciendo respuestas coherentes y contextualmente relevantes).
    - El LLM contiene **base de conocimiento** (vastas cantidades de conocimiento de sus datos de entrenamiento).
    - El LLM **comprende y mantiene el contexto**.
  - **Componente de gestión del diálogo**: es crucial para mantener conversaciones coherentes y atractivas; además, se mantiene un **flujo natural** en las conversaciones.
  - **Integración con aplicaciones:** en la integración con aplicaciones el modelo se accede a través de APIs desde aplicaciones o servicios.

La gestión del diálogo

- **La gestión de diálogo involucra:**
  - mantener la pista de la **historia de la conversación** (incluyendo prompts y respuestas del sistema previos),
  - gestión del **estado del diálogo actual** de la conversación (incluye preferencias de usuario e interacciones previas),
  - **gestión de diálogo** (mantiene la historia de diálogo, gestiona estrategias de conversación y decide las respuestas apropiadas),
  - **aprendizaje de políticas** (crea caminos felices para las conversaciones, guiando interacciones hacia resultados positivos).

Diferencias entre un modelo lingüístico de propósito general y un chatbot conversacional inteligente

- **Un modelo lingüístico de propósito general** (**MLPG**) – p.ej: GPT 4, BERT - está diseñado como un modelo de lenguaje versátil capaz de realizar una amplia gama de tareas, incluyendo la generación de texto, resumen, traducción, y más.
- Un MLPG no está diseñado específicamente para interacciones conversacionales.
- Un MPLG es capaz de gestionar consultas complejas en diversos ámbitos y producir respuestas coherentes y contextualmente pertinentes en función de la información recibida.
- Un MLPG carece de gestión de diálogo integrada: no gestiona de forma inherente el flujo de la conversación, ni realiza un seguimiento del historial de diálogo.
- Cada prompt se trata por el MLPG de forma independiente, sin el contexto de interacciones anteriores.
    - El modelo genera respuestas basadas únicamente en la última entrada sin tener en cuenta el historial de la interacción.

Diferencias entre un modelo lingüístico de propósito general y un chatbot conversacional inteligente

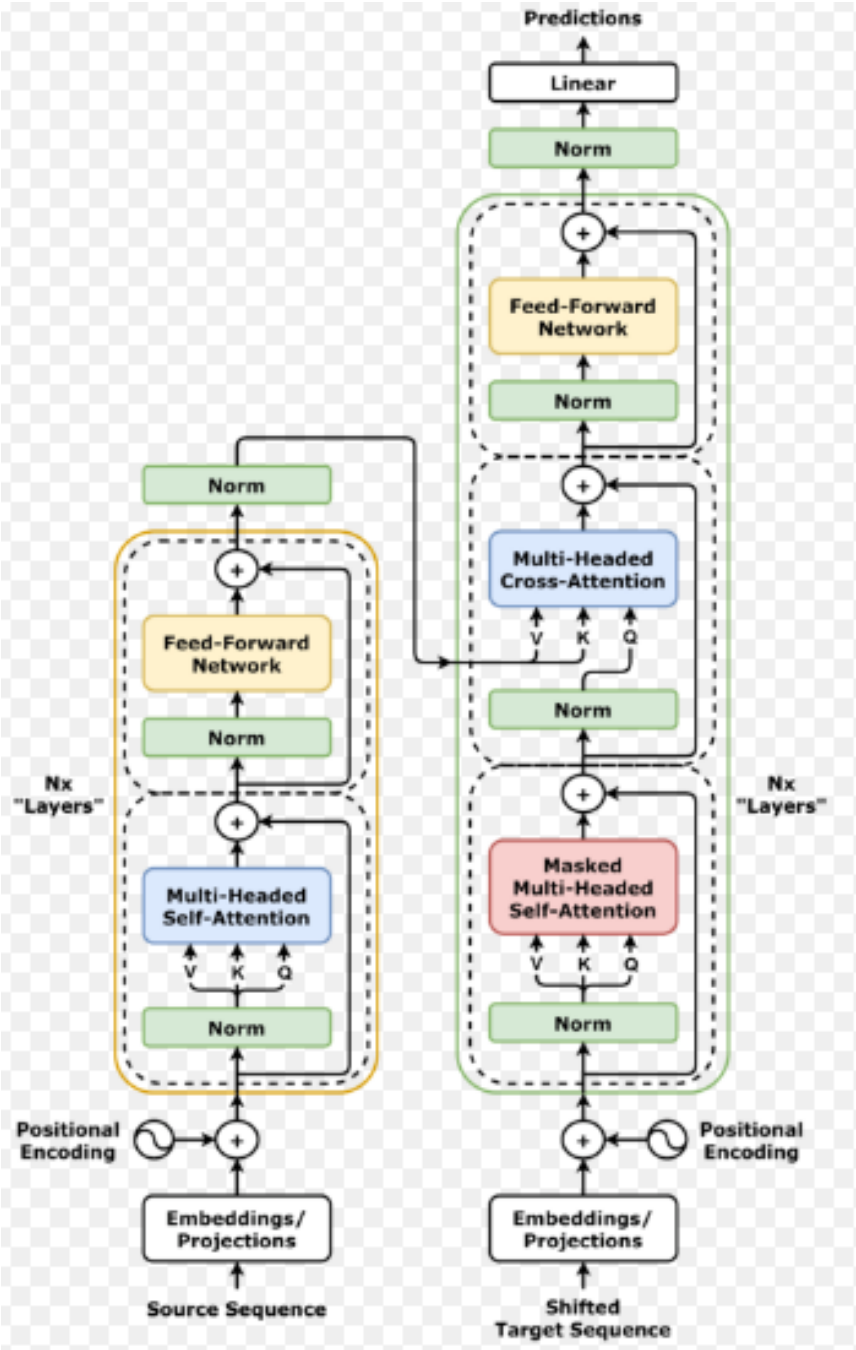
- Una **IA conversacional** – p.ej. chatGPT 4, Copilot - está diseñada específicamente para participar en conversaciones.
- Una IA conversacional Incorpora **estrategias de gestión del diálogo** para mantener el contexto y la coherencia a lo largo de distintas interacciones (i.e. prompts).
  - Su principal objetivo es proporcionar respuestas similares a las humanas en un formato conversacional.
- Una IA conversacional puede seguir el **estado de la conversación**, reconocer las intenciones del usuario y **gestionar el contexto** en múltiples intercambios.
  - Esta capacidad le permite responder adecuadamente basándose en interacciones anteriores, lo que hace que las conversaciones sean más coherentes y atractivas.

Diferencias entre un modelo lingüístico de propósito general y un chatbot conversacional inteligente

- ChatGPT 4 involucra a los usuarios de una manera más dinámica, permitiendo intercambios de ida y vuelta en los que el modelo puede hacer preguntas aclaratorias o proporcionar información de seguimiento basada en las aportaciones previas del usuario.

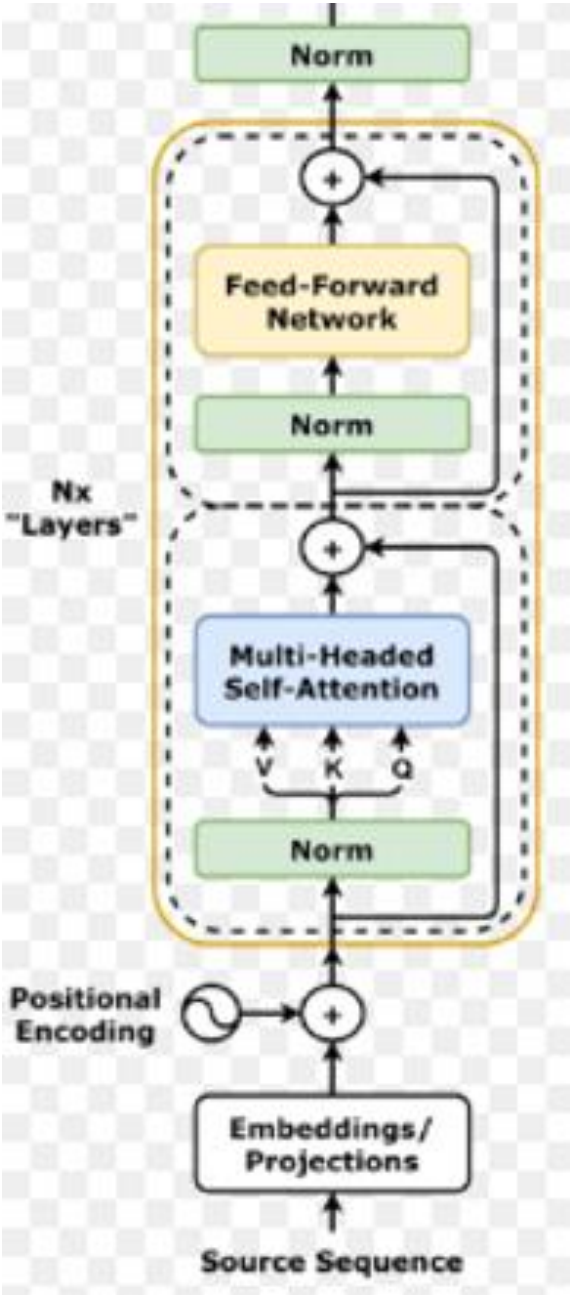
Arquitectura de transformers

- Esta arquitectura permite el **procesamiento paralelo de datos** y es especialmente efectiva para tareas de procesamiento del lenguaje natural (NLP).
- Utiliza **mecanismos de autoatención** que permiten al modelo enfocarse en diferentes partes del texto de entrada, lo que mejora la comprensión del contexto y las relaciones entre palabras.
- Las componentes clave de un transformer son: Codificador y decodificador.
  - El **codificador** procesa la entrada y
  - el **decodificador** genera la salida basada en la **representación vectorial** creada por el codificador.
- Un transformer genera texto **prediciendo** una palabra a la vez, usando el contexto proporcionado por las palabras anteriores.



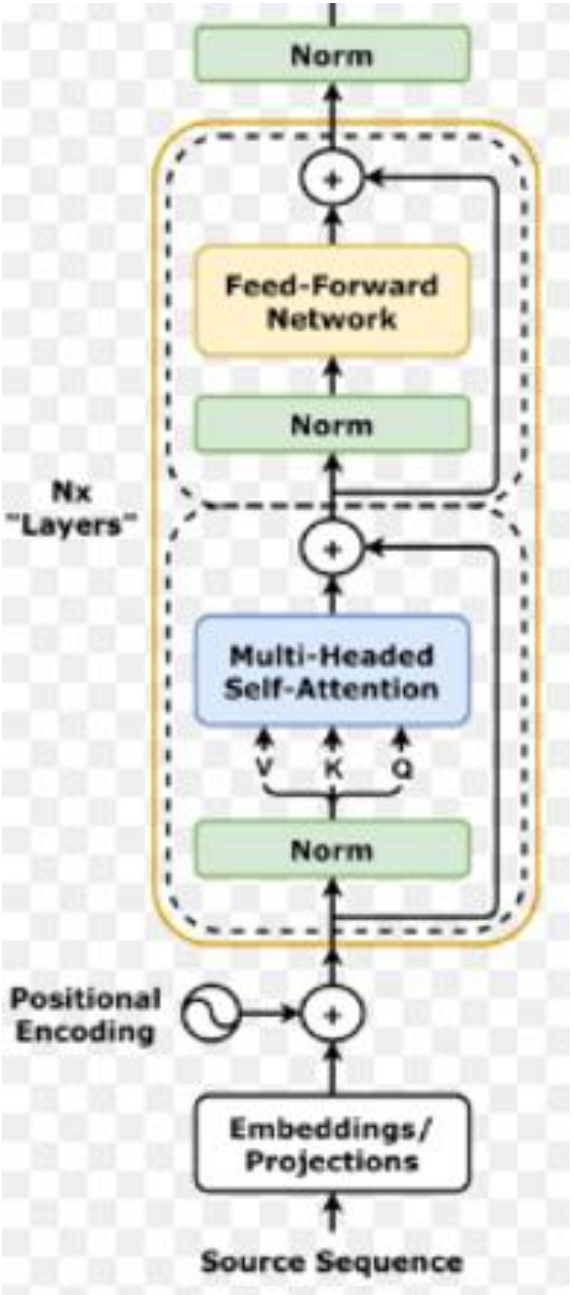
El codificador

- Los codificadores permiten **procesar** y **entender** secuencias de texto.
- Un codificador va a comprender el prompt; pero no hace solo esto, porque a partir del texto de entrada generará **representaciones contextuales** que capturan el significado y las relaciones entre las palabras.
- El **resultado de un codificador** es una **serie de vectores** que representan cada palabra en el contexto del prompt.
- Un codificador está compuesto por varias **capas apiladas**, cada una de las cuales realiza funciones específicas.



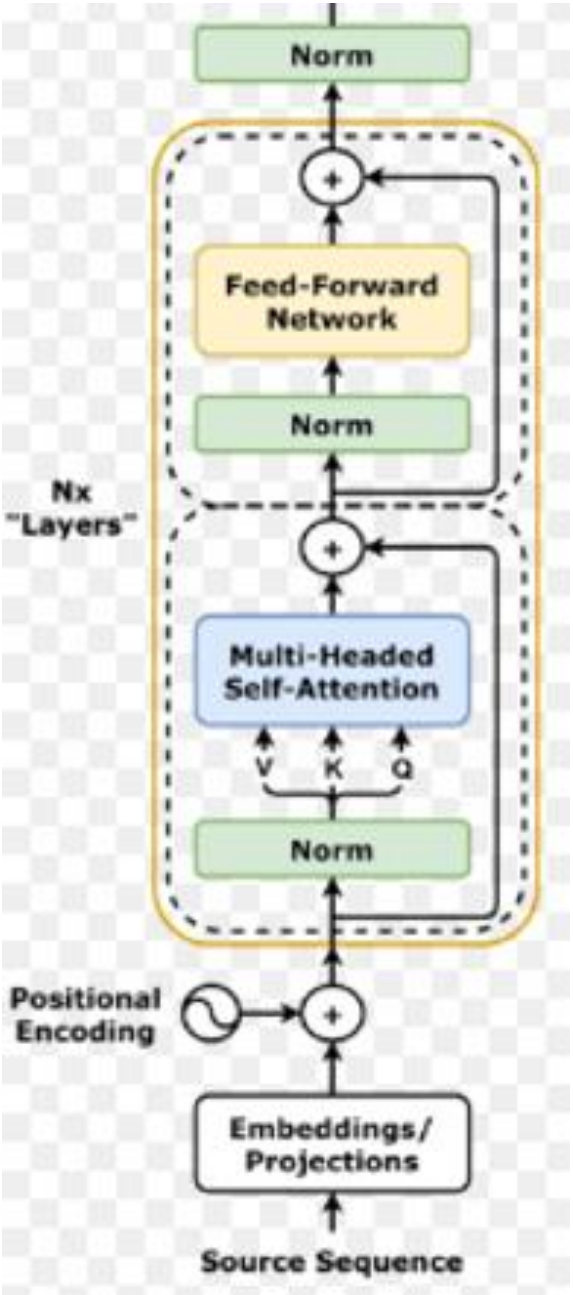
Fases del codificador

1. **Entrada de Texto**: El texto original es tokenizado y convertido en incrustaciones que representan matemáticamente cada palabra.
   - Cada token se convierte a un **vector numérico** usando una técnica de incrustación (embedding).
2. **Aplicación de codificación posicional:** La técnica de codificación posicional agrega información sobre la posición de cada palabra en la secuencia,
   - permitiendo al modelo entender el orden y las relaciones temporales entre las palabras.
   - La codificación posicional es un **vector de tamaño fijo** representando las posiciones relativas de los tokens dentro de la secuencia.



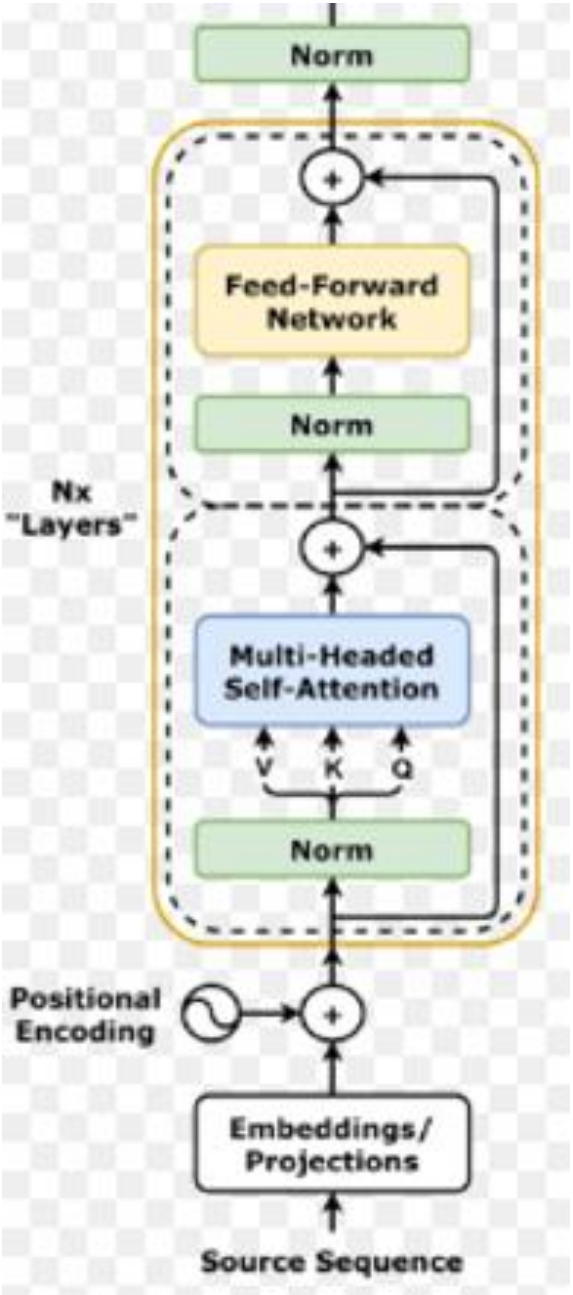
Fases del codificador

3. **Cálculo de autoatención**: La **capa de autoatención** permite que el modelo evalúe la relevancia de cada palabra en la entrada con respecto a las demás palabras.
   - Utiliza un **mecanismo de atención** que asigna diferentes pesos a las palabras según su importancia contextual.
   - Los pesos miden la importancia de cada token en relación con los demás tokens de la secuencia.
   - **Aclaraciones**:
     - Los pesos para un token miden las similitudes del token con todos los demás tokens y estos pesos suman 1.
     - Si un peso de atención es alto entre dos tokens, indica que la relación entre los tokens es muy importante pare entender el contexto en ese punto de la secuencia.
     - Si un peso de atención es bajo, significa que esa relación es menos relevante en el contexto actual.



Fases del codificador

3. **La capa de alimentación directa**: transforma la salida de la capa de autoatención utilizando una red neuronal que aplica funciones no lineales para **mejorar la capacidad del modelo para aprender patrones complejos**.
   - Se genera una nueva representación para cada palabra, **enriqueciendo aún más su contexto.**
   - **Aclaraciones:**
     - La capa de alimentación directa principalmente enriquece y refina las relaciones creadas por la capa de auto-atención.
     - Su objetivo es fortalecer y profundizar la comprensión de las relaciones existentes capturadas.
     - Este trabajo es para facilitar aún más la identificación de patrones más complejos.
     - Para identificar patrones más complejos están las N capas que combinan auto-atención con alimentación directa.



El codificador

- **Ejemplo**: consideremos la oración: La inteligencia artificial está revolucionando el futuro del trabajo humano rápidamente
  - **Relaciones importantes en la auto-atención:**
    - "inteligencia artificial": Una relación directa y clave que establece el sujeto.
    - "revolucionando el futuro": Relación entre la acción y el objeto directo, que da contexto sobre el impacto.
    - "futuro del trabajo": Identifica específicamente el área afectada.
    - "trabajo humano": Añade un nivel adicional de especificidad sobre quién se ve afectado.
  - **Refinamiento de relaciones en la alimentación directa:**
    - Profundización de "revolucionando el futuro": Captura la magnitud y la dirección del cambio, sugiriendo innovación y disrupción.
    - Especificación de "futuro del trabajo humano": Integra el concepto de transformación en el ámbito laboral, afectando a las habilidades y roles humanos.

El codificador

- Si hubiera dos capas de auto-atención y alimentación directa, la segunda capa toma las salidas refinadas de la primera capa como su nueva entrada.
  - En cierto sentido, la segunda capa trata estas representaciones enriquecidas como unidades para identificar relaciones o patrones más complejos.
- **Ejemplo**: en la oración: "El médico que trató al paciente diagnosticó la enfermedad rara,"
  - La primera capa identifica las relaciones importantes: médico-trató, trató-paciente, diagnosticó-enfermedad, enfermedad-rara.
  - La segunda capa ve la relación médico-trató como una unidad. Además, ve “diagnosticó la enfermedad rara” como un evento clínico crucial. Además, integra la relación médico-paciente con la acción de diagnóstico.

El codificador

- Los codificadores también pueden ser utilizados en diversas tareas dentro del procesamiento del lenguaje natural (NLP), tales como:
  - **Clasificación de Texto**: Determinar categorías o etiquetas para documentos.
  - **Reconocimiento de Entidades Nombradas**: Identificar nombres propios y otros términos relevantes dentro del texto.
  - **Análisis Sentimental**: Evaluar el tono emocional del contenido.

El codificador

- Los codificadores producen varias **representaciones intermedias**:
  - **Vectorización**: Las palabras se transforman usando el mecanismo de embedding en vectores en un espacio multidimensional, donde las distancias entre los vectores reflejan **similitudes semánticas**.
  - **Representaciones Contextuales**: Cada palabra se representa no solo por su significado individual, sino también por su **relación con otras palabras** en la oración.
  - **Salidas para el Decodificador**: La salida final del codificador se pasa al decodificador, proporcionando un **contexto rico** para generar respuestas o traducciones.

El decodificador

- Los **decodificadores** en la arquitectura de los **transformers** son componentes esenciales que permiten generar salidas a partir de las representaciones contextuales producidas por los codificadores.

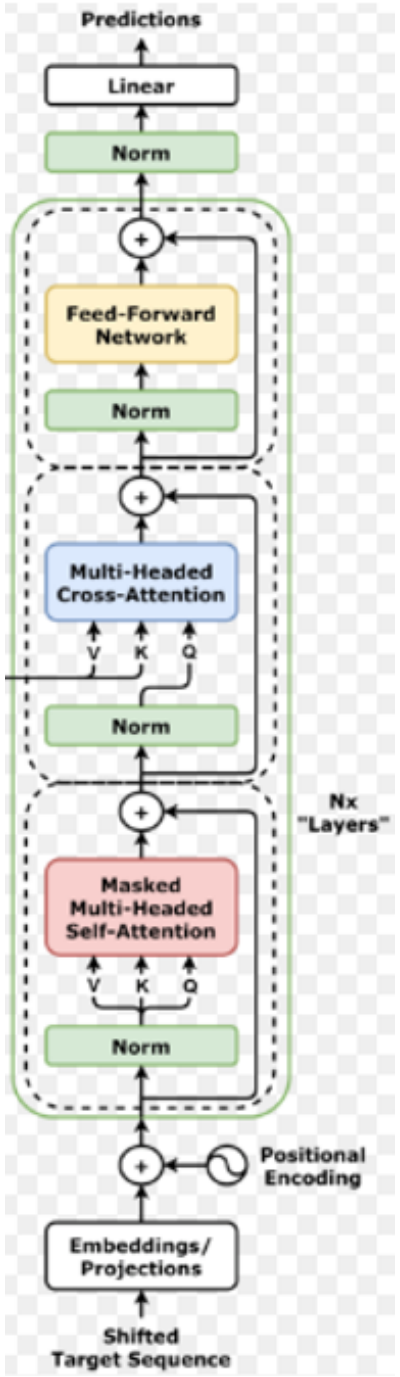
- Los decodificadores son utilizados en diversas tareas dentro del procesamiento del lenguaje natural (NLP), tales como:

  - **Generación de Texto**: Creación automática de contenido basado en un contexto dado.

  - **Traducción Automática**: Conversión de texto de un idioma a otro.

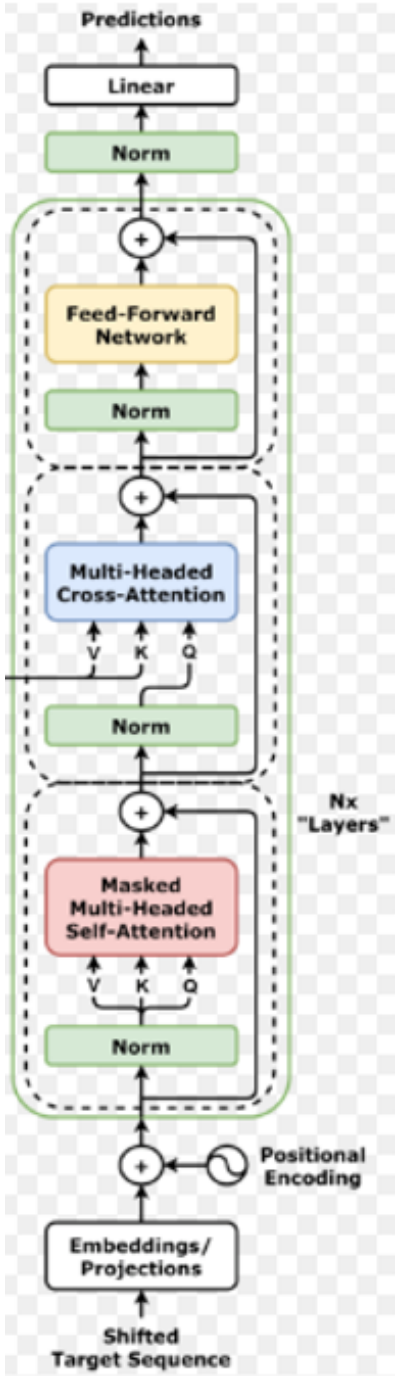
  - **Respuestas a Preguntas**: Generación de respuestas coherentes y relevantes basadas en consultas específicas.

- Un decodificador en un modelo de transformer está compuesto por varias **capas apiladas**, cada una con funciones específicas.



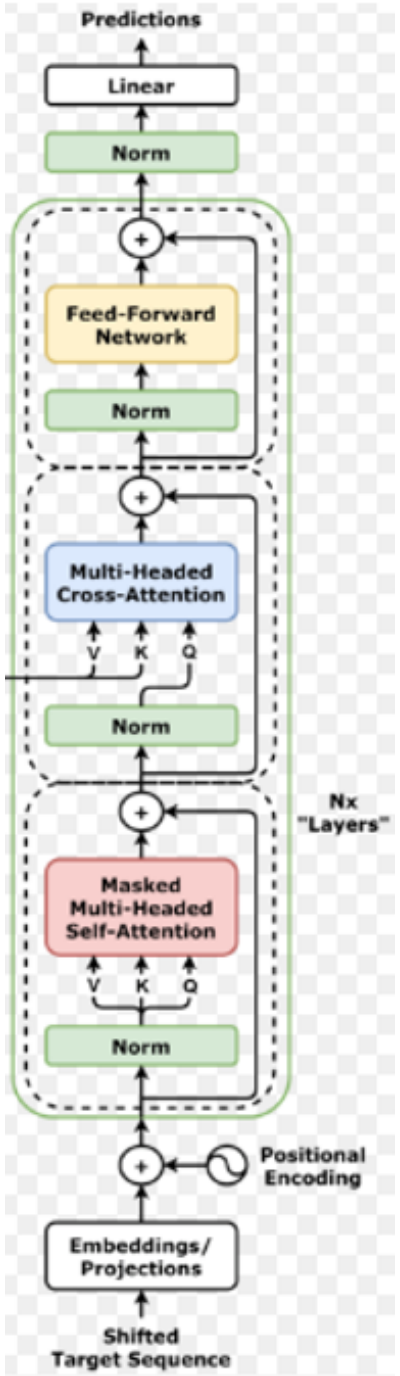
Capas del decodificador

- **Capa de autoatención**: Esta capa permite que el decodificador preste atención a las palabras generadas previamente en la secuencia de salida.
  - Esto es crucial para mantener la coherencia y el contexto a medida que se genera texto.
  - Produce representaciones que reflejan la importancia de cada palabra generada hasta el momento.
- **Capa de atención cruzada**: Esta capa permite al decodificador acceder a la representación generada por el codificador.
  - Utiliza la información contextual de la entrada para influir en la generación de la salida.
  - Facilita la integración del contexto de entrada en el proceso de generación, mejorando la relevancia y precisión de las respuestas.



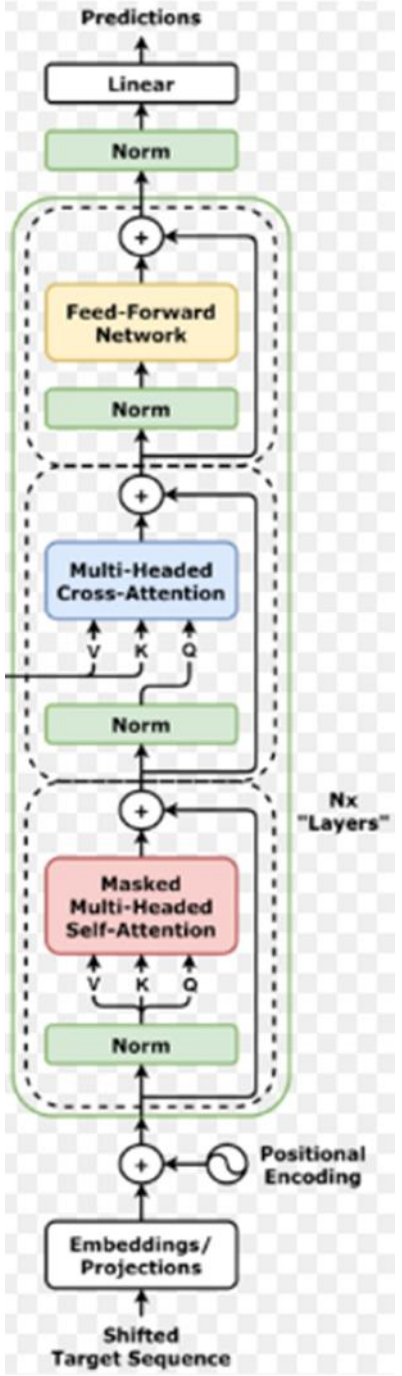
Capas del decodificador

- **Capa de alimentación directa**: esta capa transforma la salida de las capas anteriores utilizando una red neuronal completamente conectada.
  - Se usa para enriquecer adicionalmente las representaciones antes de pasar a la siguiente capa.
- **Codificación posicional**: se utiliza una **codificación posicional** para permitir que el decodificador comprenda el orden de las palabras generadas.
  - Esto es esencial para mantener la coherencia en las oraciones.



Capas del decodificador

Capa líneal: toma la representación final del contexto generada por las capas anteriores y la transforma en una puntuación para cada palabra posible del vocabulario. Elige entre ellas la palabra de mayor puntuación.



Fases del decodificador

1. **Entrada Inicial:** La entrada consiste en los embeddings de las palabras generadas hasta ese momento, junto con información posicional.

2. **Cálculo de Autoatención**: La capa de autoatención evalúa las palabras generadas hasta ahora para determinar su relevancia mutua.

3. **Atención Cruzada:** Se realiza atención cruzada utilizando las representaciones del codificador para integrar información relevante del contexto original.

    - En otras palabras: la capa de atención cruzada combina la información contextual del codificador con la información de contexto del propio decodificador, creando así una salida coherente y contextualizada.

Fases del codificador

4. **Transformación a través de la Capa Feed-Forward:** La salida se transforma mediante una red neuronal para obtener representaciones finales.
5. **Generación Final:** La última capa del decodificador aplica una transformación lineal y una función softmax para producir probabilidades sobre el vocabulario, permitiendo seleccionar la palabra más probable como salida.

Decodificador

- La **generación de la siguiente palabra** de salida en el decodificador ocurre en la capa final del decodificador, después de que todas las capas de auto-atención, atención cruzada y alimentación directa hayan procesado la información.
  - Después de todas las capas de auto-atención, atención cruzada y alimentación directa, se obtiene una representación final del contexto del texto generado hasta ese punto.
  - Esta representación se pasa por una capa de salida (usualmente una capa densa) que transforma la representación final en una probabilidad para cada posible siguiente token.
  - Se genera así una lista de palabras posibles en el vocabulario junto con sus probabilidades (a esto se le llama generación de logits)
  - El token con la mayor probabilidad es seleccionado como la siguiente palabra.

Decodificador

- **Ejemplo**: si el decodificador está en el proceso de generar la siguiente palabra después de "El médico", podría producir una lista de palabras con probabilidades como:
  - "diagnosticó" (0.7)
  - "trató" (0.2)
  - "observó" (0.05)
  - "recetó" (0.05)
- En este caso, "diagnosticó" tendría la mayor probabilidad y sería seleccionada como la siguiente palabra en la secuencia.

Observaciones sobre los transformers

- la **arquitectura de transformers** se utiliza tanto para el **entrenamiento** del modelo como para **procesar prompts** y **generar respuestas**.
  - La parte de procesar prompts y generar respuestas ya fue explicada en detalle.
- **Durante el entrenamiento**, se utiliza la arquitectura de transformers para procesar grandes volúmenes de texto.
  - Esta arquitectura permite al modelo **aprender** patrones del lenguaje, estructuras gramaticales y contextos semánticos a partir de datos textuales masivos.
  - El **modelo se entrena** inicialmente con un enorme conjunto de datos textuales, donde aprende a predecir la siguiente palabra en una secuencia.
  - Este proceso implica el uso de mecanismos de atención y capas de codificación que son parte integral de la arquitectura de transformers.

Entrenamiento de un transformer

- Durante el **entrenamiento de un transformer**, se utiliza un proceso de **aprendizaje supervisado** que funciona de la siguiente manera:
  - **Entrada**: Se proporciona una oración completa, pero se oculta la última palabra.
  - **Predicción**: El modelo intenta predecir la palabra que falta basándose en el contexto de las palabras anteriores.
  - **Comparación**: La predicción del modelo se compara con la palabra real que se ocultó.
  - **Cálculo de Error:** Se calcula el error o la diferencia entre la predicción y la palabra real.
  - **Retropropagación**: Si la predicción es incorrecta, el error se utiliza para ajustar los pesos del modelo mediante un proceso llamado retropropagación.
    - ❑ Este ajuste ayuda al modelo a mejorar sus predicciones futuras.
- Este ciclo se repite millones de veces con diferentes oraciones y datos, lo que permite que el modelo se vuelva más preciso y efectivo en sus predicciones.

Entrenamiento de un transformer

- Otra forma de entrenar un transformer es usando un proceso llamado **enmascaramiento de lenguaje.** Es una técnica comúnmente de **aprendizaje supervisado** utilizada durante el entrenamiento de modelos como BERT. Consiste de las siguientes etapas:
  - **Enmascaramiento**: Algunas palabras en la oración se reemplazan con un token especial (por ejemplo, [MASK]).
  - **Predicción**: El modelo intenta predecir las palabras que faltan basándose en el contexto de las palabras circundantes.
  - **Comparación**: Se compara la predicción del modelo con las palabras reales.
  - **Cálculo de Error**: El error se calcula y se utiliza para ajustar los pesos del modelo.
- **Ejemplo**: si tenemos la oración "El médico [MASK] al paciente", el modelo intentará predecir la palabra faltante "trató". E
- sta técnica permite que el modelo aprenda relaciones bidireccionales y entienda mejor el contexto completo de la oración.

Entrenamiento de un transformer

- Los **patrones aprendidos** durante el entrenamiento son fundamentales para predecir la siguiente palabra en una secuencia.
  - Al entrenarse con vastos volúmenes de texto, el modelo capta patrones comunes en la estructura del lenguaje, las relaciones entre palabras y el contexto semántico.
  - Utiliza este conocimiento para hacer predicciones precisas al generar texto.
- Cuando se le presenta un contexto, el modelo utiliza los patrones previamente aprendidos para estimar la probabilidad de cada posible siguiente palabra.
  - Esta capacidad de entender y aplicar patrones contextuales es lo que permite a los transformers generar texto coherente y relevante.
  - Es como tener una memoria vasta y aplicarla al vuelo para decidir qué palabra sigue, basándose en lo que "ha visto" antes.

Entrenamiento de un transformer

- Los **patrones lingüísticos aprendidos** no se almacenan como en una base de datos tradicional; en cambio, se reflejan en los valores de los millones de parámetros del modelo.
  - Durante el entrenamiento, el modelo ajusta estos parámetros para capturar las relaciones y patrones del lenguaje.
  - Esto permite que el modelo genere texto y haga predicciones basadas en su comprensión del contexto, todo sin necesidad de una base de datos externa.
  - Es como si todos los conocimientos adquiridos estuvieran codificados en su "cerebro" en forma de parámetros ajustados.

Entrenamiento de un transformer

- El modelo ajusta sus parámetros en función de las comparaciones entre las predicciones y las etiquetas correctas (extraídas del texto),
  - lo que permite que aprenda las relaciones contextuales y patrones lingüísticos de manera efectiva.
  - Esta metodología permite al modelo entender y generar texto de forma coherente y contextualizada.
  - Durante el entrenamiento, el transformer ajusta los parámetros de sus capas de auto-atención y alimentación directa para aprender patrones lingüísticos y contextuales.

Limitaciones de un bot conversacional de texto inteligente

Sesgos

- **Descripción**: Los modelos de IA pueden heredar sesgos de los datos con los que fueron entrenados. Estos sesgos pueden estar relacionados con género, raza, cultura, y más.
- **Impacto**: Las respuestas pueden reflejar y perpetuar prejuicios, llevando a resultados injustos o inapropiados.
- **Ejemplo**: Si un modelo está entrenado con datos que tienen un sesgo de género, puede generar respuestas que refuercen estereotipos.

Alucinaciones

- **Descripción**: El término "alucinaciones" en IA se refiere a cuando el modelo genera información que es incorrecta o no basada en datos reales.
- **Impacto**: Puede llevar a la desinformación y a la pérdida de confianza en el modelo.
- **Ejemplo**: Si se le pregunta por un hecho histórico, el modelo podría inventar detalles que no son ciertos.

Limitaciones de un bot conversacional de texto inteligente

Repeticiones

- **Descripción**: Los modelos de IA pueden caer en patrones de repetición, especialmente si no reciben suficiente variabilidad en las entradas.
- **Impacto**: Las respuestas pueden volverse monótonas y parecer menos naturales o interesantes.
- **Ejemplo**: Un modelo puede repetir frases como "Eso suena interesante" frecuentemente, haciendo que las interacciones se sientan repetitivas.

Capacidades Limitadas de Resolución de Problemas

- **Descripción**: Los bots pueden tener dificultades para resolver problemas complejos o no estructurados que requieren pensamiento crítico.
- **Impacto**: Esto puede resultar en la necesidad de intervención humana para resolver ciertos problemas.

Limitaciones de un bot conversacional de texto inteligente

Comprensión del Contexto Limitada

- **Descripción**: A veces, el bot puede tener dificultades para mantener el contexto de una conversación prolongada o compleja.
- **Impacto**: Esto puede llevar a respuestas fuera de contexto o irrelevantes.

Ambigüedad e Interpretación

- **Descripción**: Los bots pueden luchar con preguntas ambiguas o frases con múltiples significados.
- **Impacto**: Esto puede resultar en respuestas incorrectas o confusas para el usuario.

Limitaciones de un bot conversacional de texto inteligente

Dependencia de Entradas de Calidad

- **Descripción**: Los bots son tan buenos como los datos con los que han sido entrenados.
- **Impacto**: Datos sesgados o de baja calidad pueden llevar a respuestas sesgadas o inapropiadas.