Curso Corto en Transformers y LLMs

A. M. Álvarez Meza, Ph. D,
D. A. Pérez, M.Eng.

Universidad Nacional de Colombia
Signal Processing and Recognition Group - SPRG

August 28, 2025

Contenido

Contenido

Objetivo General

Brindar una comprensión de **mecanismos de atención y arquitecturas Transformer/LLM** con enfoque específico en el análisis de criticidad en redes CHEC.

Alcance y Capacidades

- Fortalecer la **toma de decisiones** basada en datos.
- Abordar y resolver problemas complejos de manera innovadora.
- Capacitar para **reentrenar y actualizar soluciones de IA** previamente desarrolladas en la sinergia CHEC-UNAL.

Aspectos Clave

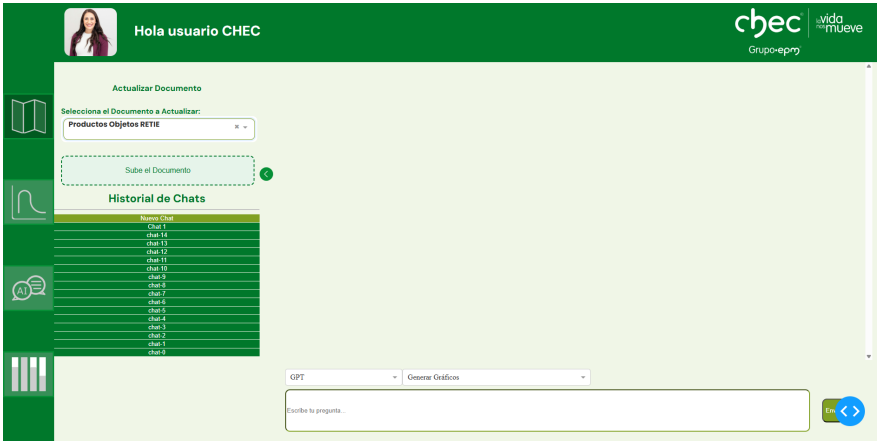
- **Duración Total:** 26 horas
  - 16 horas de clase directa (4 semanas).
  - 10 horas de asesoría extra-clase.
- **Equipo Docente:** Profesores de la UNAL.

Cronograma

Cronograma curso corto en transformers y LLMs para el análisis de criticidad

	Semana 1				Semana 2				Semana 3				Semana 4			
Tema	1h	2h	3h	4h	1h	2h	3h	4h	1h	2h	3h	4h	1h	2h	3h	4h
Atención/Transformer	X	X														
TabNet			X	X												
LLMs					X	X	X	X								
Fine-tuning									X	X						
Contextos											X	X				
Práctica/Reentreno.			X	X			X	X							X	X
Asesorías							X	X	X	X	X	X	X	X	X	X

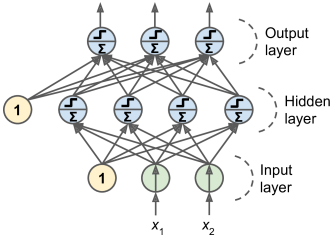
Resultados Esperados

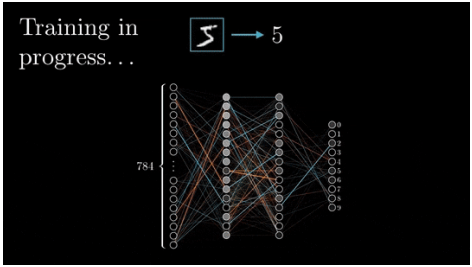


Manejo y reentrenamiento de modelos con datos de media tensión de CHEC.

Contenido

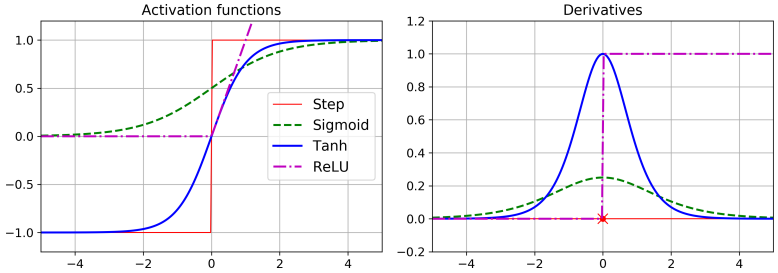
MLP and Backpropagation





La señal fluye en una dirección, y el error se propaga en la opuesta (Backpropagation).

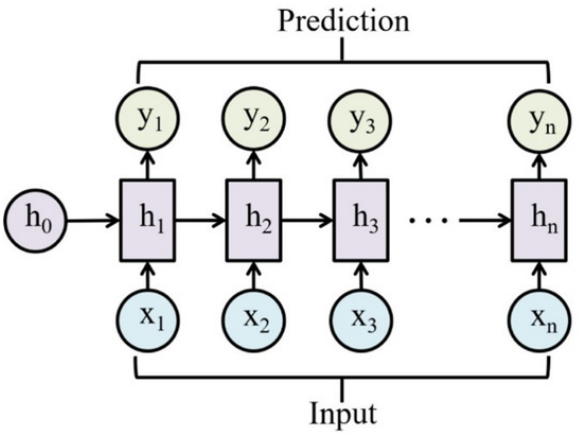
MLP: Funciones de activación II



- Si **encadenamos varias transformaciones lineales**, el resultado sigue siendo una única **transformación lineal**.
- Por tanto, si no se introducen funciones **no lineales** entre capas, incluso una red profunda es equivalente a una sola capa lineal.
- Esto puede visualizarse claramente en el Playground de TensorFlow, donde una red con múltiples capas pero activación lineal *no logra separar datos no linealmente* .

Contenido

Redes Neuronales Recurrentes (RNN)

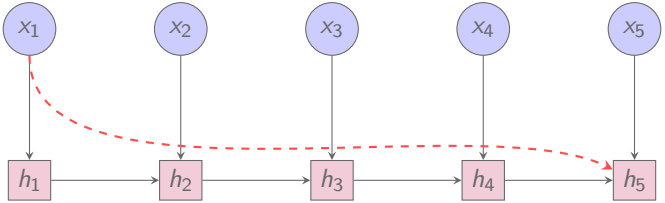


- **Concepto Clave:** Procesar secuencias paso a paso.
- La salida en un instante de tiempo t depende de la entrada actual (x_t) y de la **"memoria"** del paso anterior (h_{t-1}).
- Este **estado oculto (h_t)** actúa como un resumen del pasado.
- Ideal para datos donde el orden importa (texto, series de tiempo).

Limitaciones de las RNN

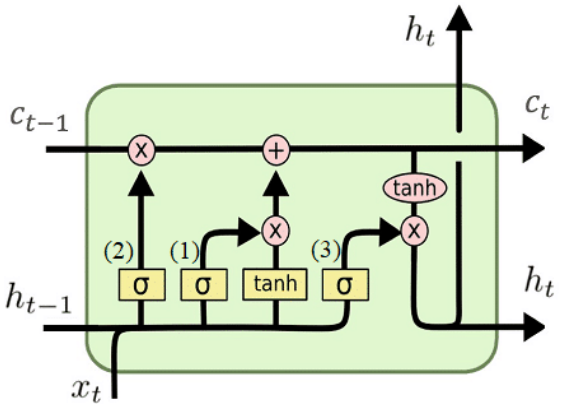
Problema: Memoria a Corto Plazo

El flujo de información a través de muchos pasos degrada la señal. El gradiente se desvanece (o explota), dificultando el aprendizaje.



- **Consecuencia:** La red **olvida** el contexto de palabras o eventos lejanos en la secuencia.
- **La pregunta clave:** ¿Cómo podemos controlar selectivamente qué recordar y qué olvidar?

Memoria a Largo y Corto Plazo (LSTM)



LSTM
(Long-Short Term Memory)

- **Innovación:** Un "carril de memoria" dedicado, el **Estado de la Celda** (C_t).
- El flujo de información en este carril es controlado por tres **compuertas**:
  - **Olvido (Forget):** Decide qué borrar de la memoria.
  - **Entrada (Input):** Decide qué información nueva añadir.
  - **Salida (Output):** Decide qué parte de la memoria usar.

Limitaciones de las LSTM

Ventaja: Solución a la Memoria

Las compuertas permiten a la red mantener dependencias a largo plazo, solucionando en gran medida el problema del gradiente desvanecido.

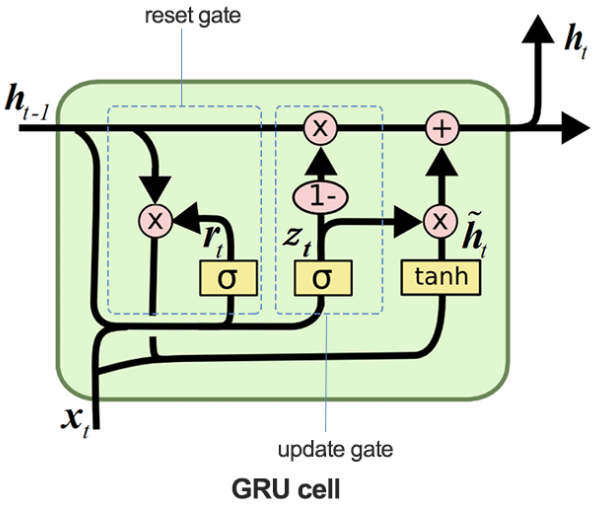
Limitación Heredada: Procesamiento Secuencial

A pesar de su poder, una LSTM **debe** procesar la secuencia paso a paso. El cálculo del instante t depende del resultado del instante $t-1$.

Esto impide la paralelización.

Procesar secuencias muy largas es lento.

3. GRU: Una Versión Simplificada y Eficiente



GRU cell

- **Concepto Clave:** Una LSTM más simple y computacionalmente más barata.
- Fusiona el estado de la celda y el estado oculto en uno solo (h_t).
- Usa solo dos compuertas:
  - **Actualización (Update):** Combina las funciones de olvido y entrada. Decide cuánto del pasado mantener.
  - **Reseteo (Reset):** Decide cuánto del pasado ignorar.

Limitaciones Generales

El Gran Logro

Las arquitecturas con compuertas (LSTM y GRU) demostraron que **controlar el flujo de información** es clave para manejar secuencias.

El Obstáculo Insalvable: La Secuencialidad

TODAS las arquitecturas recurrentes comparten la misma debilidad fundamental: procesan la información en orden.

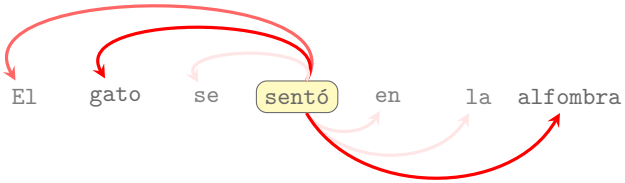
¿Y si pudiéramos procesar todos los elementos de la secuencia a la vez, entendiendo sus relaciones sin un orden estricto?

$\rightarrow$ **¡Esa es la idea detrás del Transformer!**

Contenido

Mecanismos de Atención: La Intuición

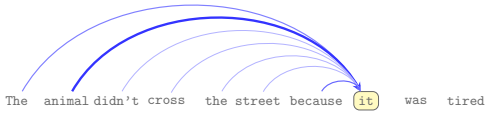
- **Problema:** En una secuencia larga, ¿qué palabras son más importantes para entender el contexto actual?
- **Idea Clave:** Permitir que el modelo **aprenda a "enfocar"** su atención en las partes más relevantes de la entrada para tomar una decisión.
- Es un mecanismo que asigna **pesos de importancia** a cada elemento de la secuencia.



Self-Attention

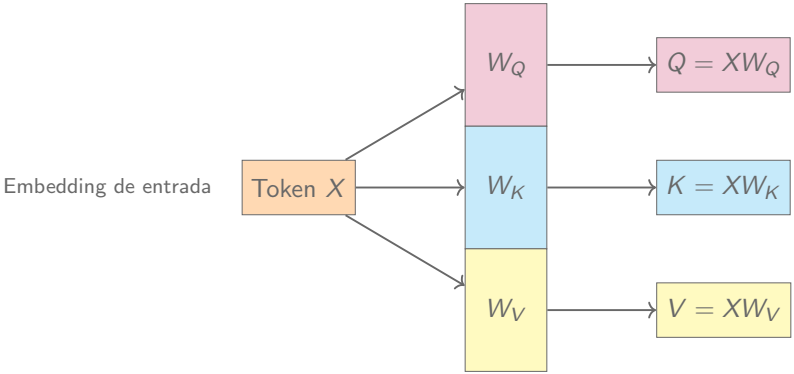
El Salto Conceptual: Rompiendo la Secuencialidad

En lugar de procesar palabra por palabra como una RNN, el Self-Attention permite que una secuencia **se analice a sí misma en su totalidad y de forma simultánea**.



- Para entender la palabra *"it"*, la red aprende a enfocarse en *"The animal"*.
- Cada palabra se relaciona con **todas las demás** en paralelo.
- **Ventaja clave:** Se eliminan las dependencias secuenciales, lo que **permite paralelización**.
- En este ejemplo se visualiza solo la atención hacia atrás, pero en modelos bidireccionales, cada palabra puede atender a todas las demás.

El "Cómo" del Self-Attention: Query, Key, Value (Q, K, V)



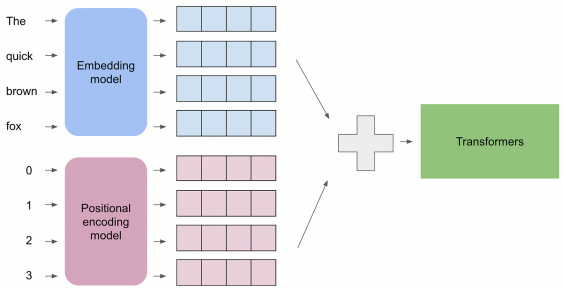
El Problema del Orden: Positional Encoding

La Debilidad del Self-Attention

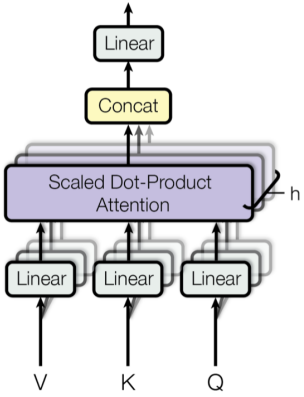
Al procesar todo a la vez, el modelo **pierde la noción del orden** de las palabras. *"El rey venció al siervo"* vs *"El siervo venció al rey"* serían indistinguibles.

La Solución: Positional Encoding

Se **inyecta** un vector con información de posición a cada palabra antes de que entre al modelo. Este vector le "dice" a la red dónde está ubicada cada palabra.



Multi-Head Attention: Múltiples Puntos de Vista



- **Idea:** Una sola atención podría centrarse en un solo tipo de relación (ej. sintáctica).
- **Multi-Head** ejecuta el mecanismo de Self-Attention **múltiples veces en paralelo**, cada una aprendiendo un tipo diferente de relación.
- Es como tener un comité de "expertos" que analizan la frase desde diferentes ángulos y luego combinan sus conclusiones.

Estructura de un Bloque Transformer

Core of the Transformer is Self Attention as in the original paper: "**Attention** is all you need".

arxiv.org/pdf/1706.03762.pdf

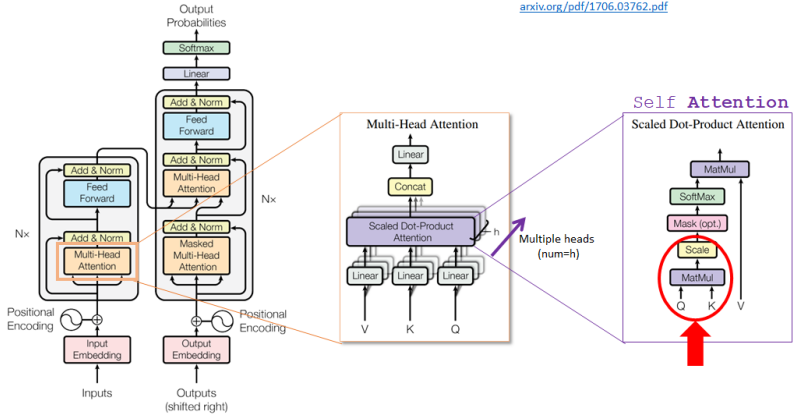
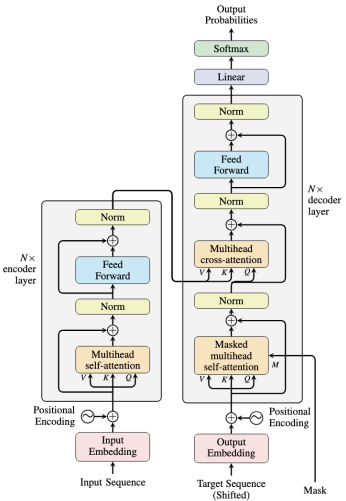


Figure 1: The Transformer - model architecture.

La Arquitectura Completa: Encoder y Decoder



Encoder (Lector)

Apila N bloques Transformer. Su trabajo es **leer y entender** la secuencia de entrada, generando una representación rica en contexto.

Decoder (Escritor)

También apila N bloques. Su trabajo es **generar** la secuencia de salida, prestando atención a lo que ya ha generado y a la representación del Encoder.

El Límite del Transformer: ¿Qué pasa con los datos tabulares?

Los LLMs brillan en texto, pero sufren con las tablas.

Los datos tabulares son fundamentalmente diferentes del lenguaje natural.

En el Texto:

- El orden importa (sintaxis).
- Relaciones semánticas globales.
- Tokens homogéneos (palabras).

En Tablas:

- El orden de las columnas es arbitrario.
- Relaciones locales y específicas.
- Features heterogéneas (numéricas, categóricas...).

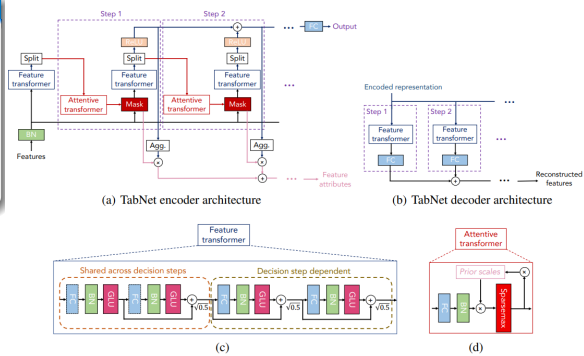
Aplicar un Transformer directamente a una tabla es ineficiente y a menudo da malos resultados.

Introducción a TabNet: Atención para Tablas

La Idea de TabNet

En lugar de que todas las features interactúen con todas, TabNet usa un **mecanismo de atención secuencial** para seleccionar las features más importantes en cada paso de decisión.

- Imita la forma en que los modelos de árboles seleccionan features en cada nodo.
- Usa **máscaras de atención dispersas** para elegir un subconjunto de features en cada paso.
- Esto lo hace **interpretable por diseño**: podemos ver qué features usó en cada



En cada paso, una máscara de atención selecciona un subconjunto de features.

Comparativa: TabNet vs. XGBoost

TabNet	XGBoost
Modelo: Red Neuronal Profunda (Deep Learning).	**Modelo:** Ensamblado de Árboles de Decisión (Gradient Boosting).
Interpretabilidad: Integrada (*local y global*) a través de las máscaras de atención.	**Interpretabilidad:** Post-hoc, a través de herramientas como SHAP o Feature Importance.
Preprocesamiento: Menos sensible. Puede aprender representaciones de features (end-to-end).	**Preprocesamiento:** Muy sensible. Requiere una cuidadosa ingeniería de características.
Rendimiento: Altamente competitivo, a veces superior, especialmente con muchas features.	**Rendimiento:** A menudo el estándar de la industria (SOTA). Extremadamente rápido y robusto.
Cómputo: Requiere GPU para ser eficiente. Más lento de entrenar.	**Cómputo:** Muy eficiente en CPU. Rápido de entrenar.

Muchas Gracias!

Contacto:

Diego Armando Pérez Rosero

dieaaperezros@unal.edu.co