Transformers and Large Language Models - Winter School - PUCP

Parte I: Introducción a los *Transformers*

Sebastián Vallejo Vera [1]

August 18, 2025

[1] Dept. of Political Science – Western University

Índice

La evolución de los métodos de NLP

Métodos en evolución

Históricamente, el Procesamiento Natural del Lenguaje (NLP) se ha enfocado en medir alguna característica latente en un conjunto de textos (corpus).

- Bag-of-Words: *Wordfish* (Slapin and Proksch, 2008).
- Embeddings: ubicación ideológica de partidos, i.e., *party-embeddings* (Rheault and Cochrane, 2020).
- Transformers: modelos supervisados para la clasificación de texto (Timoneda and Vallejo Vera, 2025*b*).

Más recientemente, se ha incrementado el interés en Generative Large Language Models (LLM).

- Aplicaciones: dimensiones latentes, e.g., ideología (Kato and Cochrane, 2025; Wu et al., 2023); anotadores (Timoneda and Vallejo Vera, 2025*c*).
- Limitaciones: sesgos derivados de señales partidistas (Vallejo Vera and Driggers, 2025); sesgos del lenguaje (Walker and Timoneda, 2024).

Métodos en evolución

Históricamente, el Procesamiento Natural del Lenguaje (NLP) se ha enfocado en medir alguna característica latente en un conjunto de textos (corpus).

- Bag-of-Words: *Wordfish* (Slapin and Proksch, 2008).
- Embeddings: ubicación ideológica de partidos, i.e., *party-embeddings* (Rheault and Cochrane, 2020).
- Transformers: modelos supervisados para la clasificación de texto (Timoneda and Vallejo Vera, 2025*b*).

Más recientemente, se ha incrementado el interés en Generative Large Language Models (LLM).

- Aplicaciones: dimensiones latentes, e.g., ideología (Kato and Cochrane, 2025; Wu et al., 2023); anotadores (Timoneda and Vallejo Vera, 2025*c*).
- Limitaciones: sesgos derivados de señales partidistas (Vallejo Vera and Driggers, 2025); sesgos del lenguaje (Walker and Timoneda, 2024).

Métodos en evolución

Históricamente, el Procesamiento Natural del Lenguaje (NLP) se ha enfocado en medir alguna característica latente en un conjunto de textos (corpus).

- Bag-of-Words: *Wordfish* (Slapin and Proksch, 2008).
- Embeddings: ubicación ideológica de partidos, i.e., *party-embeddings* (Rheault and Cochrane, 2020).
- Transformers: modelos supervisados para la clasificación de texto (Timoneda and Vallejo Vera, 2025*b*).

Más recientemente, se ha incrementado el interés en Generative Large Language Models (LLM).

- Aplicaciones: dimensiones latentes, e.g., ideología (Kato and Cochrane, 2025; Wu et al., 2023); anotadores (Timoneda and Vallejo Vera, 2025*c*).
- Limitaciones: sesgos derivados de señales partidistas (Vallejo Vera and Driggers, 2025); sesgos del lenguaje (Walker and Timoneda, 2024).

Transformers

¿Qué son los Transformers?

- Los *Transformers* son una arquitectura de *machine-learning* utilizada en modelos basados en lenguaje.

¿Qué son los Transformers?

- Los *Transformers* son una arquitectura de *machine-learning* utilizada en modelos basados en lenguaje.
- Podemos pensar en los *Transformers* como los motores que impulsan los modelos de *machine-learning* y mejoran (significativamente) su rendimiento.

¿Qué son los Transformers?

- Los *Transformers* son una arquitectura de *machine-learning* utilizada en modelos basados en lenguaje.
- Podemos pensar en los *Transformers* como los motores que impulsan los modelos de *machine-learning* y mejoran (significativamente) su rendimiento.
- Los *Transformers* son la arquitectura detrás de la mayoría (¿todos?) de los modelos de lenguaje de última generación (SOTA): BERT, RoBERTa, Llama, etc.

¿Conozco algún Transformers?

¿He estado expuesto antes a esta arquitectura?

Conoces algunos Transformers

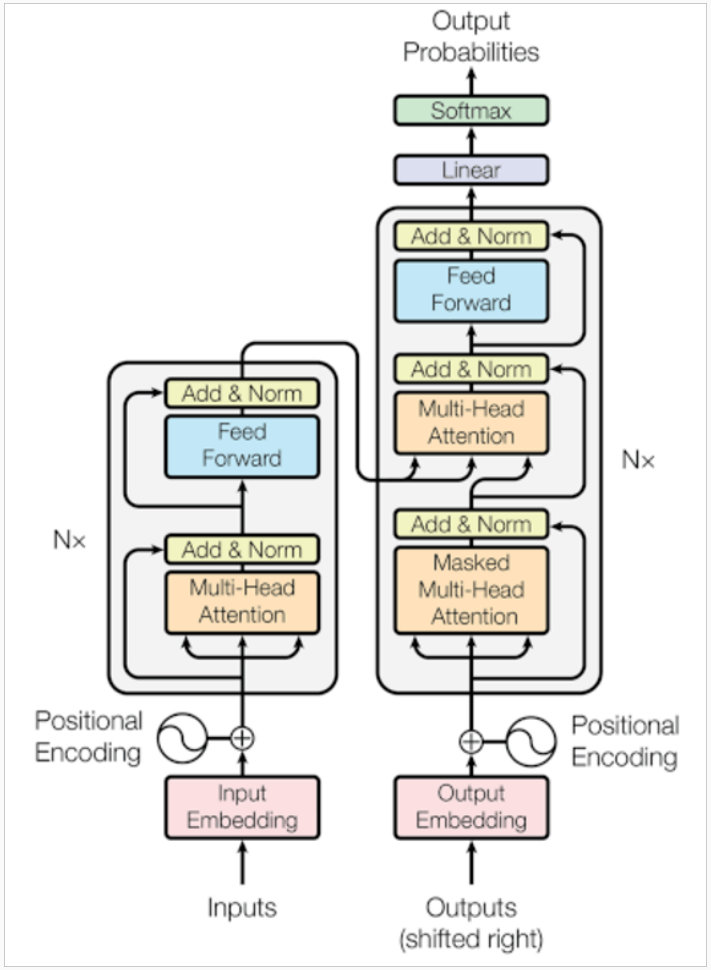
¡Sí los conoces!

- Google Translate (y predicción).
- ChatGPT, Grok 😲, etc.
- GitHub Copilot.
- Generación automática de subtítulos en redes sociales.

GPT = *Generative Pre-trained Transformers*

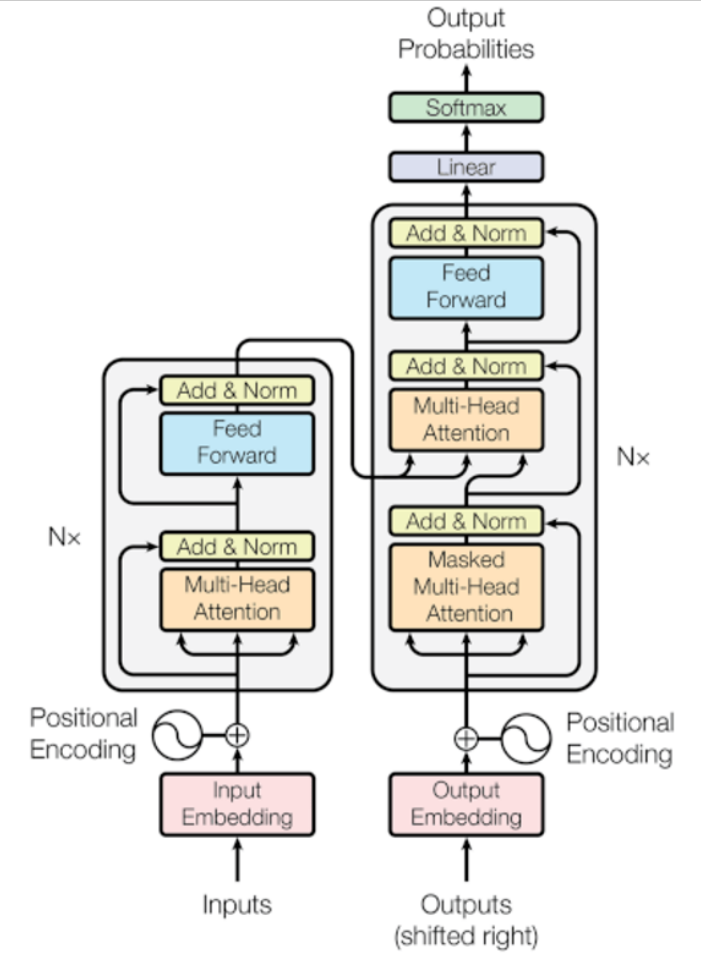
¿Cómo Funcionan los Transformers?

La Arquitectura Encoder - Decoder



- Es complicado...

La Arquitectura Encoder - Decoder



- Es complicado... pero podemos desglosarlo en partes.

Mecanismo del Encoder

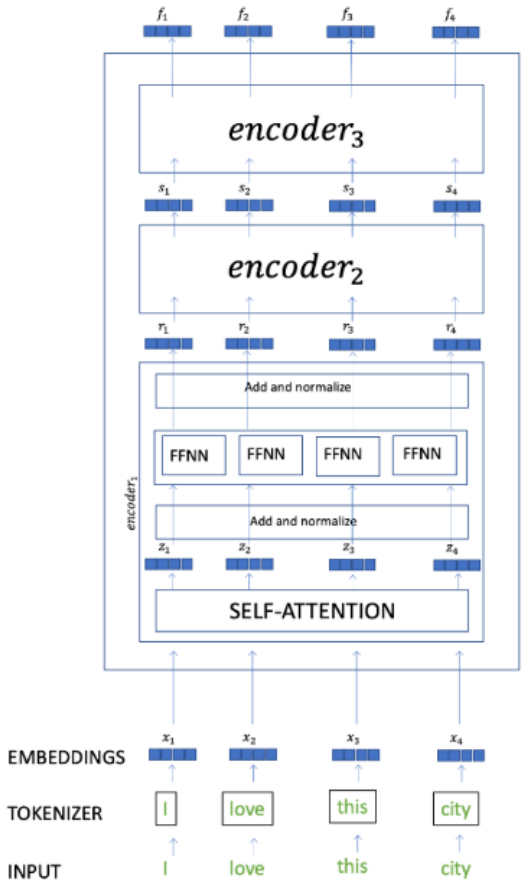


Figure 1: Diagram of a three-stack encoder of a Transformers model. Input text is tokenized and given an initial embedding (vectorized representation) simplified in our figure as x_1 through x_4. The initial embeddings are transformed as they enter the first $encoder_1$. In it, the self-attention mechanism updates the embeddings (z_1 through z_4), which are then passed through a feed-forward neural network. They exit the encoder as a more accurate set of embeddings (r_1 through r_4). The process is repeated for all encoders in the neural network. For example, pre-trained BERT-base models use 12 encoder layers.

Masking

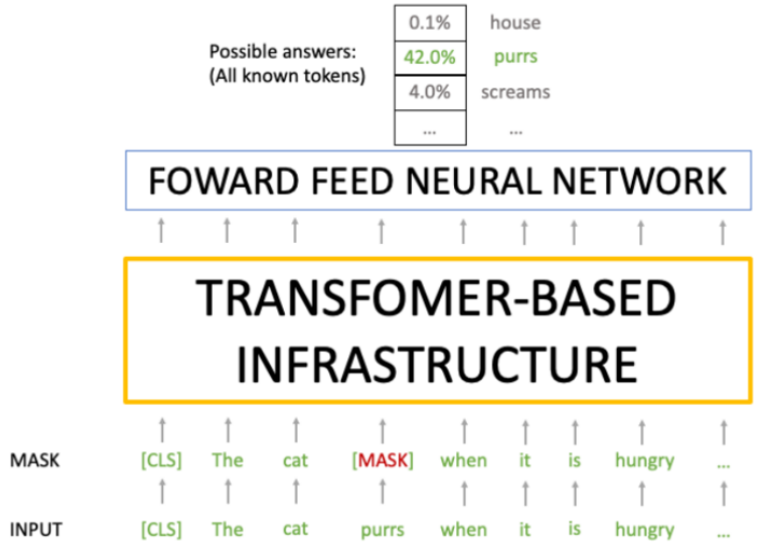
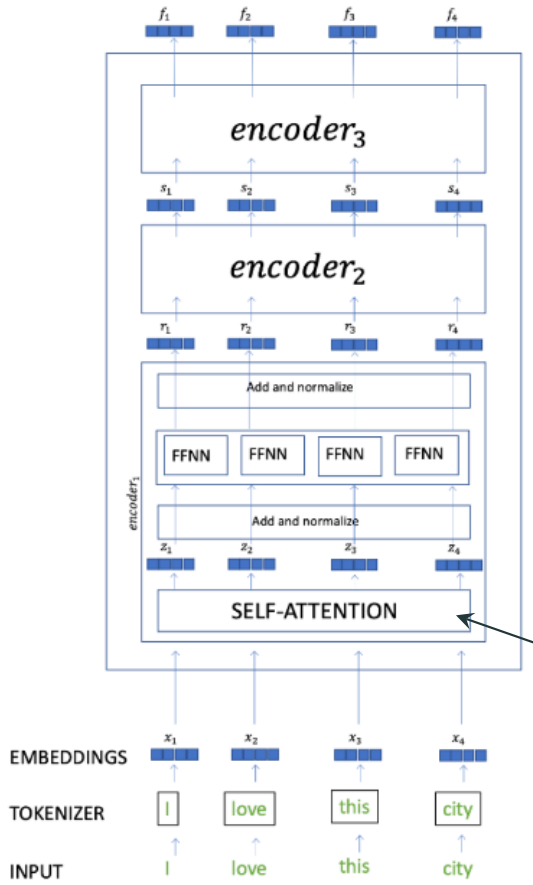


Figure 2: Representation of the Masked Language Modeling. Transformer models randomly mask 15% of tokens. After running the corpus through the Transformers and neural-network architecture, it asks the model to predict the masked word. Transformer models then calculate loss and the required gradient changes to optimize the model's weights and obtain better representations.

Esto se llama un *procedimiento cloze*, desarrollado en Taylor (1953)... el único artículo no relacionado con ciencias de la computación citado en Vaswani et al. (2017).

Mecanismo del Decoding



Attention Is All You Need

Figure 1: Diagram of a three-stack encoder of a Transformers model. Input text is tokenized and given an initial embedding (vectorized representation) simplified in our figure as x_1 through x_4. The initial embeddings are transformed as they enter the first $encoder_1$. In it, the self-attention mechanism updates the embeddings (z_1 through z_4), which are then passed through a feed-forward neural network. They exit the encoder as a more accurate set of embeddings (r_1 through r_4). The process is repeated for all encoders in the neural network. For example, pre-trained BERT-base models use 12 encoder layers.

Mecanismo de Self-Attention

El mecanismo de self-attention (‘Attention Is All You Need’) es el componente revolucionario de la arquitectura (Vaswani et al., 2017).

- Cada entrada consiste en matrices de ‘queries’ y ‘keys’. En la oración ‘A Maríale gustan los libros’, para darle significado a la palabra ‘María’ (query) observamos toda la oración y encontramos la palabra que más está relacionada, en este caso, ‘gusta’ (key).

Mecanismo de Self-Attention

El mecanismo de self-attention (‘Attention Is All You Need’) es el componente revolucionario de la arquitectura (Vaswani et al., 2017).

- Cada entrada consiste en matrices de ‘queries’ y ‘keys’. En la oración ‘A Maríale gustan los libros’, para darle significado a la palabra ‘María’ (query) observamos toda la oración y encontramos la palabra que más está relacionada, en este caso, ‘gusta’ (key).
- El mecanismo de self-attention estimará la distancia (producto punto) entre la representación vectorial de cada consulta con la representación vectorial ya dada de cada clave.

Mecanismo de Self-Attention

El mecanismo de self-attention (‘Attention Is All You Need’) es el componente revolucionario de la arquitectura (Vaswani et al., 2017).

- Cada entrada consiste en matrices de ‘queries’ y ‘keys’. En la oración ‘A Maríale gustan los libros’, para darle significado a la palabra ‘María’ (query) observamos toda la oración y encontramos la palabra que más está relacionada, en este caso, ‘gusta’ (key).
- El mecanismo de self-attention estimará la distancia (producto punto) entre la representación vectorial de cada consulta con la representación vectorial ya dada de cada clave.
- Nota que, aunque los tokens pueden ser los mismos, las representaciones serán diferentes (ya que provienen de capas anteriores).

Mecanismo de Self-Attention

El mecanismo de self-attention ('Attention Is All You Need') es el componente revolucionario de la arquitectura (Vaswani et al., 2017).

- Cada entrada consiste en matrices de 'queries' y 'keys'. En la oración 'A Maríale gustan los libros', para darle significado a la palabra 'María' (query) observamos toda la oración y encontramos la palabra que más está relacionada, en este caso, 'gusta' (key).
- El mecanismo de self-attention estimará la distancia (producto punto) entre la representación vectorial de cada consulta con la representación vectorial ya dada de cada clave.
- Nota que, aunque los tokens pueden ser los mismos, las representaciones serán diferentes (ya que provienen de capas anteriores).
- Matemáticamente, obtenemos el producto punto de *K*(eys) y *Q*(ueries) tal que $Attention(Q, K, V) = softmax(QK^T)/\sqrt{d_k})V$ donde d es la dimensión de K y se usa como factor de escala.

Vista Neuronal del Self-Attention

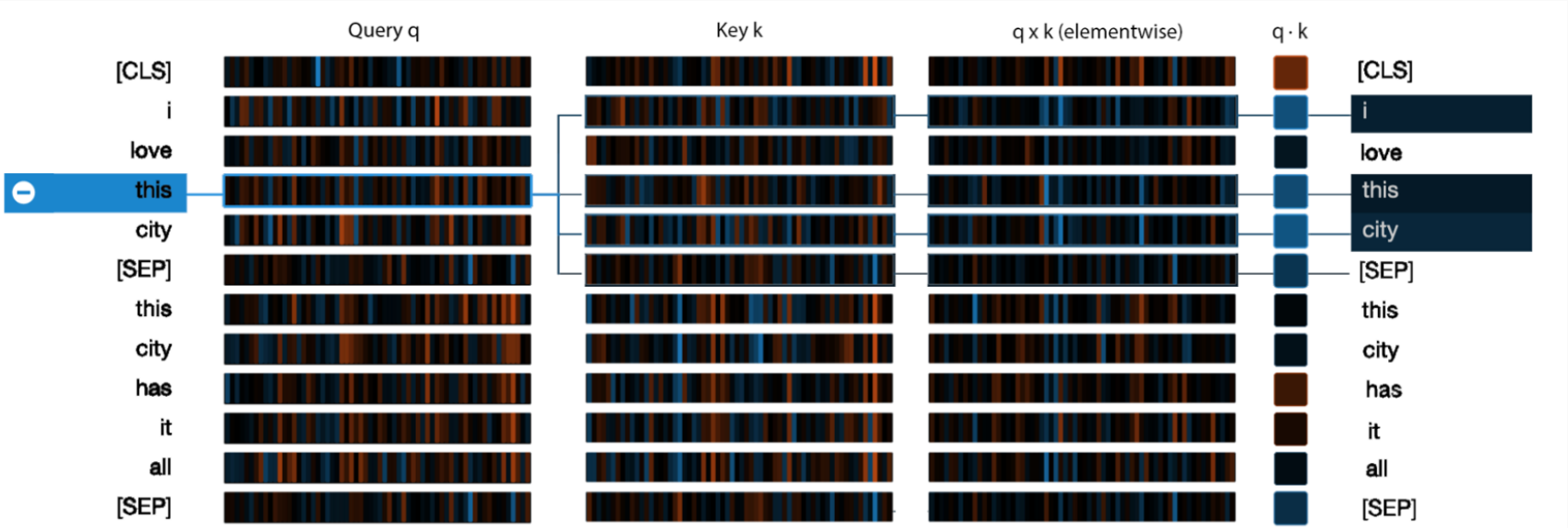


Figure F.2: Neuron view of the attention pattern at Layer 4, Head 1 of the BERT-base pretrained model. The neuron view visualizes the query and key vectors that are used to compute attention. Each color band represents a single neuron value, where color intensity indicates the magnitude and hue the sign (blue=positive, orange=negative).

La Última Capa i

Una nota final sobre self-attention es cómo los embeddings para el mismo token cambiarán según el contexto. Esta es la verdadera naturaleza de la cita “conocerás una palabra por la compañía que guarda”.

- Tome la palabra ‘bear’: puede significar el animal *oso*, un oso de peluche–*teddy bear*, o *soportar* un peso–*bear the weight*. Todos tienen diferentes significados, por lo que deben tener diferentes embeddings.
- Dado que los embeddings son representaciones numéricas de palabras, la distancia entre dos embeddings es una aproximación de la relación entre palabras.

La Última Capa ii

- En las oraciones: ‘The kid had to bear the loss of misplacing his teddy bear and his doll,’ la similitud de coseno entre oso de peluche (teddy bear) y muñeca (doll) debería ser mayor que la similitud de coseno entre oso de peluche (teddy bear) y soportar la pérdida (bear the loss). ¡Gracias al attention mechanism, esto es así! (ver Apéndice H en Timoneda and Vallejo Vera, 2025*a*)

Positioning Encoding i

- Un corpus grande no puede ser procesado en su totalidad por el proceso de encoding. Por lo tanto, se divide en una lista de vectores de tamaño 512.
- Al dividir el corpus, se mantienen elementos clave de la estructura: [CLS] y [SEP] representan el inicio y el fin de una oración, respectivamente.
- También se añade un *positioning encoding* a la lista de vectores para mantener el orden de cada token en la oración. Dado que los embeddings fluyen independientemente a través de la FFNN, el positioning encoding (un vector) ayuda a mantener el orden mientras proporciona una relación entre palabras.

Positioning Encoding ii

- El positioning encoding (PE) adopta una forma sinusoidal:

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

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

- La intuición detrás de esto es que el positioning encoding permite al modelo encontrar fácilmente la posición de cada token en un batch. También permite contar el número de pasos entre tokens: mientras mayor es la distancia, más rápido decae (más débil la relación entre estos tokens).

Positioning Encoding iii

Vaswani et al. (2017) al escribir sobre positioning encoding dice que "we chose this function because we hypothesized it would allow the model to easily learn to attend by relative positions, since for any fixed offset k, PE_{pos+k} can be represented as a linear function of PE_{pos}". Aunque no hay una prueba formal en su artículo de esta afirmación, aparentemente funciona.

¿Qué Significa Todo Esto? i

¡Es mucho! También puede ser un poco abrumador. Lo simplificaremos cuando lo implementemos, pero por ahora la pregunta más urgente debería ser: ¿qué hacemos con todo esto?

- Primero, al igual que aprendiste con los embeddings de palabras, podemos preentrenar modelos. Bueno, no 'nosotros'... personas* con mucha potencia computacional. *Por ejemplo, Google, Meta y OpenAI.
- Los modelos preentrenados son encoders: crean representaciones de tokens, *encoding* a estos con 'significado'. Estos son el SOTA en la (comprensión) de texto: BERT, RoBERTa.
- También podríamos entrenar modelos para generar texto (i.e., GPT o LLM).

¿Qué Significa Todo Esto? ii

- Los Generative Pre-Trained Transformers (GPT) o los Large Language Models (LLM) son decoders: toman alguna entrada (por ejemplo, una pregunta en una consulta de búsqueda), la interpretan (i.e., *la decodifican*) y producen una respuesta que coincide con la entrada. Estos son lo mejor en la (producción) de texto (predicción del siguiente token): Llama, OpenAI.
- O podemos combinar ambos para tareas secuenciales, como traducción (por ejemplo, T5 o BART). Estos suelen ser más lentos y menos eficientes, y para la mayoría de tareas es mejor usar modelos basados en BERT o LLMs.

Reflexiones Finales

Próximos Pasos

Hemos aprendido los mecanismos básicos que sustentan la infraestructura de los Transformers. Sin embargo, aún no sabes qué hacer con ello. Esto es lo que haremos a continuación:

1. Aprender sobre modelos preentrenados basados en BERT.
2. Aprender cómo implementar los modelos preentrenados basados en BERT.
3. Aprender sobre Large Language Models (LLMs), sus aplicaciones y limitaciones.
4. Aprender cómo implementar LLMs.

¡Gracias!

¿Preguntas? ¿Comentarios? ¿Pausa?

References

Kato, Ken and Christopher Cochrane. 2025. "KOKKAI DOC: An LLM-driven framework for scaling parliamentary representatives." *arXiv preprint arXiv:2505.07118* .

Rheault, Ludovic and Christopher Cochrane. 2020. "Word embeddings for the analysis of ideological placement in parliamentary corpora." *Political analysis* 28(1):112–133.

Slapin, Jonathan B and Sven-Oliver Proksch. 2008. "A scaling model for estimating time-series party positions from texts." *American Journal of Political Science* 52(3):705–722.

Taylor, Wilson L. 1953. ""Cloze procedure": A new tool for measuring readability." *Journalism quarterly* 30(4):415–433.

Referencias ii

Timoneda, Joan C. and Sebastián Vallejo Vera. 2025*a*. "BERT, RoBERTa or DeBERTa? Comparing Performance Across Transformer Models in Political Science Text." *The Journal of Politics* 00(00):00.

Timoneda, Joan C and Sebastián Vallejo Vera. 2025*b*. "BERT, RoBERTa or DeBERTa? Comparing Performance Across Transformers Models in Political Science Text." *The Journal of Politics* 87(1):000–000.

Timoneda, Joan C and Sebastián Vallejo Vera. 2025*c*. "Memory Is All You Need: Testing How Model Memory Affects LLM Performance in Annotation Tasks." *arXiv preprint arXiv:2503.04874* .

Vallejo Vera, Sebastián and Hunter Driggers. 2025. "Bias in LLMs as Annotators: The Effect of Party Cues on Labelling Decision by Large Language Models." *arXiv preprint arXiv:2408.15895* .

Referencias iii

Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser and Illia Polosukhin. 2017. "Attention is all you need." *Advances in neural information processing systems* 30.

Walker, Christina and Joan C Timoneda. 2024. "Identifying the sources of ideological bias in GPT models through linguistic variation in output." *arXiv preprint arXiv:2409.06043* .

Wu, Patrick Y, Jonathan Nagler, Joshua A Tucker and Solomon Messing. 2023. "Large language models can be used to estimate the latent positions of politicians." *arXiv preprint arXiv:2303.12057* .