uc3m | Universidad Carlos III de Madrid

OpenCourseWare

Procesamiento de Lenguaje Natural con Aprendizaje Profundo,

Máster en Ciencia y Tecnología Informática

Tema 5 - Transformers

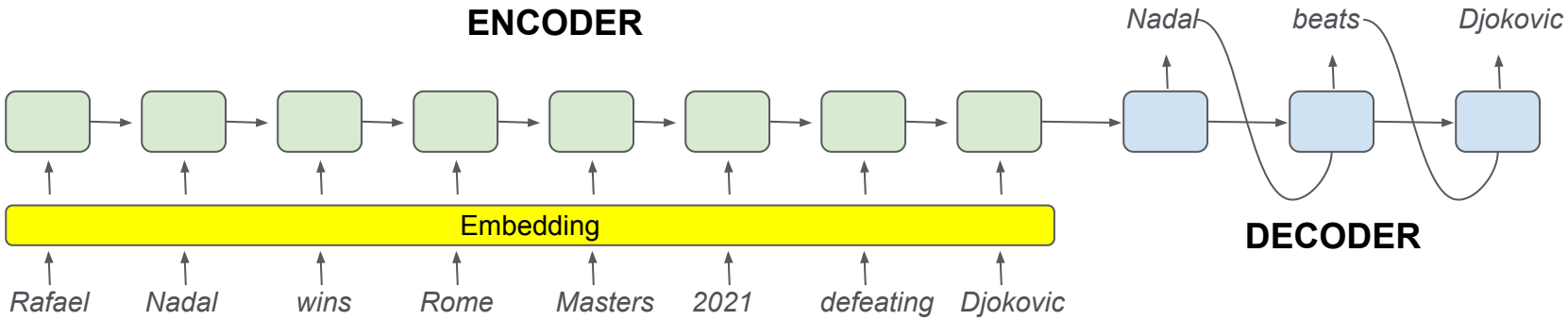
Isabel Segura Bedmar

Índice

- Seq2Seq
- Mecanismo de atención
- Primer Modelo transformer
- Tipos de transformers
- BERT
- Fine-tuning
- Conclusiones

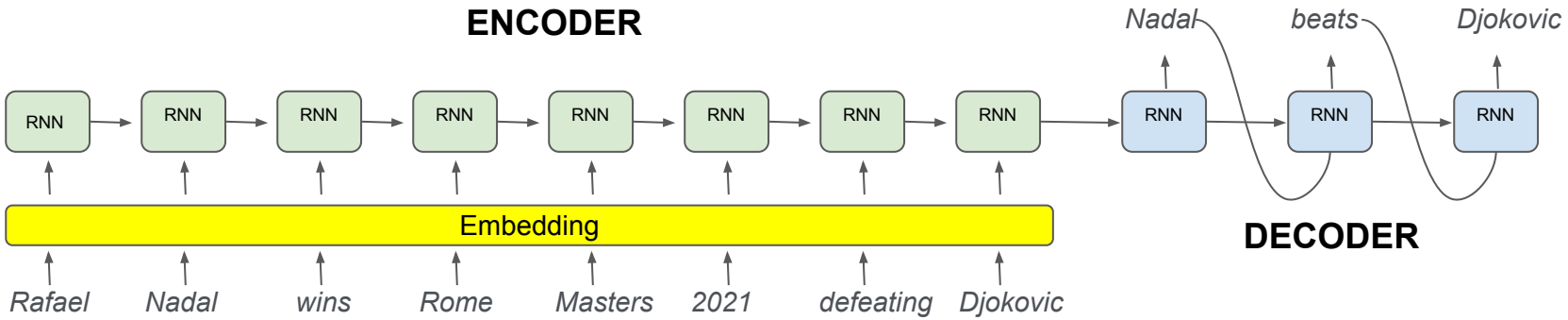
Seq2Seq

- Recibe como entrada una secuencia, y genera como salida otra secuencia.
- Arquitectura común en generación de resúmenes, traducción automática, etc.
- Compuesta por **codificador** (encoder), que lee la entrada y genera un representación intermedia, y el **decodificador** (decoder), que recibe la salida del codificador y genera la salida.



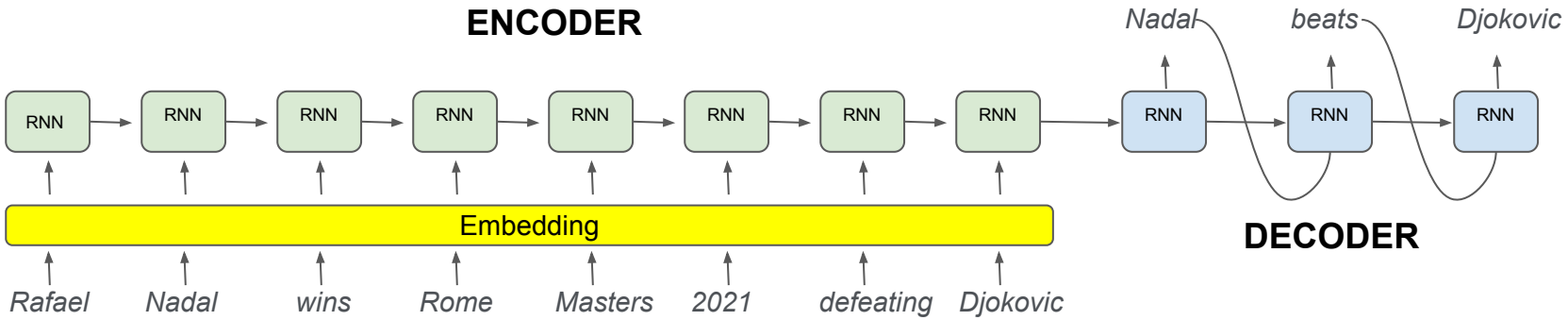
Seq2Seq

- Basadas en **redes recurrentes (RNN)**. Tanto el codificador como el decodificador están formados por una secuencia de neuronas recurrentes (por ejemplo, BiLSTM o GRU).
- La última neurona del codificador produce la entrada (estado inicial) del decodificador. En el decodificador, cada neurona produce un token de la secuencia de salida.



Seq2Seq

- Limitaciones:
  - desvanecimiento del gradiente
  - cuánto mayor sea la secuencia de entrada, más probable será que mucha de la información de los primeros tokens no esté representada en el vector que produce el codificador, y que recibe como entrada el decodificador.
  - no permiten paralelización.

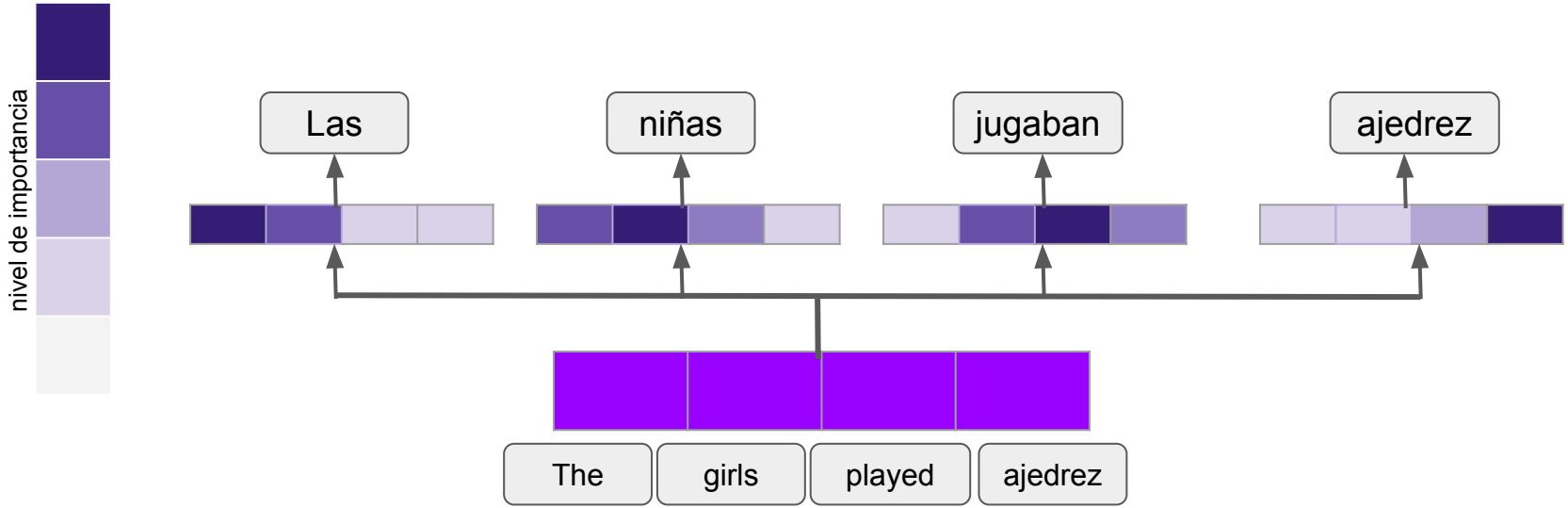


Índice

- Seq2Seq
- **Mecanismo de atención**
- Primer Modelo transformer
- Tipos de transformers
- BERT
- Fine-tuning
- Conclusiones

Solución: Seq2Seq basadas en atención.

- El **mecanismo de atención [1]** permite generar cada token en la secuencia de salida en función de los tokens más relevantes en la oración para dicho token.



Solución: Seq2Seq basadas en atención.

¿Cómo se obtiene este vector en cada paso del decodificador?,

- El codificador ya no genera un único vector de salida, sino que produce un vector por cada token de la entrada.
- El decodificador para generar cada token de salida:
  - leerá los estados generados por el codificador, y aprenderá un peso (puntuación) para cada uno de estos estados.
  - transformará dichas puntuaciones en probabilidades usando la función softmax.
  - sumará los estados ponderados con las probabilidades para obtener el vector (estado) del token actual en la salida.

Solución: Seq2Seq basadas en atención.

- Por ejemplo, para generar el tercer token de la salida (jugaban):

1) Vectores (estados) producidos por el codificador

2) Puntuaciones de atención

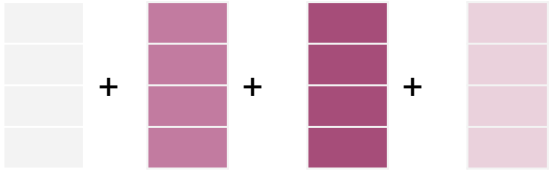
1	5	6	3

3) Probabilidades (softmax)

0.004	0.26	0.7	0.03

$$s\left(x_i\right) = \frac{e^{x_i}}{\sum_{j=1}^{n} e^{x_j}}$$

4) Sumar vectores ponderados



5) Vector para el tercer token de la salida

Pros y contras del mecanismo de atención

- RNN tienden a perder información en las secuencias largas.
- El mecanismo de atención supera esta limitación ya que todos los estados producidos por el codificador son pasados al decodificador.
- El decodificador es capaz de aprender cuáles de los estados ocultos generados por el codificador son más relevantes para cada uno de los pasos en la secuencia de salida.
- Sin embargo, este enfoque sigue teniendo una importante limitación, el **procesamiento** es todavía **secuencial**, procesando un elemento cada vez. Tanto el encoder como el decoder deben esperar a que termine el paso t-1 para comenzar con el paso t.
- Esto será especialmente ineficiente para datasets de gran tamaño.

Índice

- Seq2Seq
- Mecanismo de atención
- **Primer Modelo transformer**
- Tipos de transformers
- BERT
- Fine-tuning
- Conclusiones

Primer modelo transformer

- Vaswani, A. et al. (2017). **Attention is all you need**. *Advances in neural information processing systems, 30*.

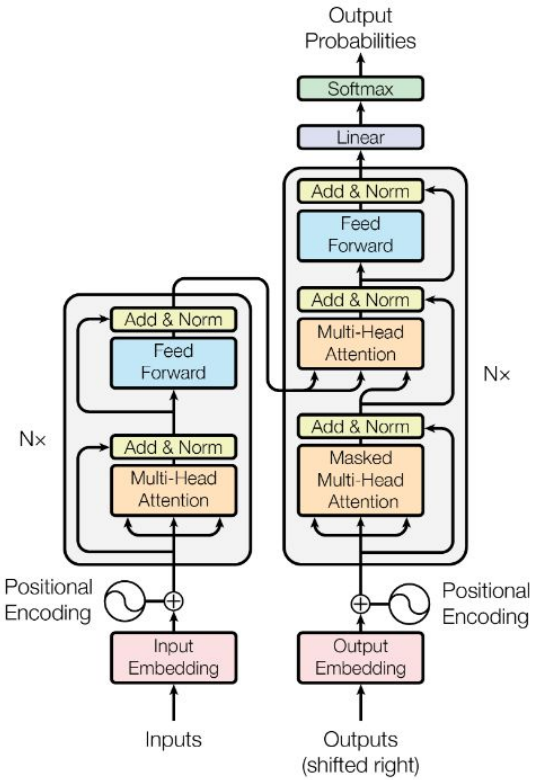


Figura tomada de Vaswani, A. et al. (2017). **Attention is all you need**. *Advances in neural information processing systems, 30.*

Primer modelo transformer

- Basado en el mecanismo auto-atención (**self-attention**):
  - Al procesar un token de la secuencia de entrada, el codificador es capaz de identificar qué otros tokens de la entrada son más relevantes para dicho token.
- No utiliza RNN, únicamente funciones de activación y sumas ponderadas.
- Eficiente y paralelizable.

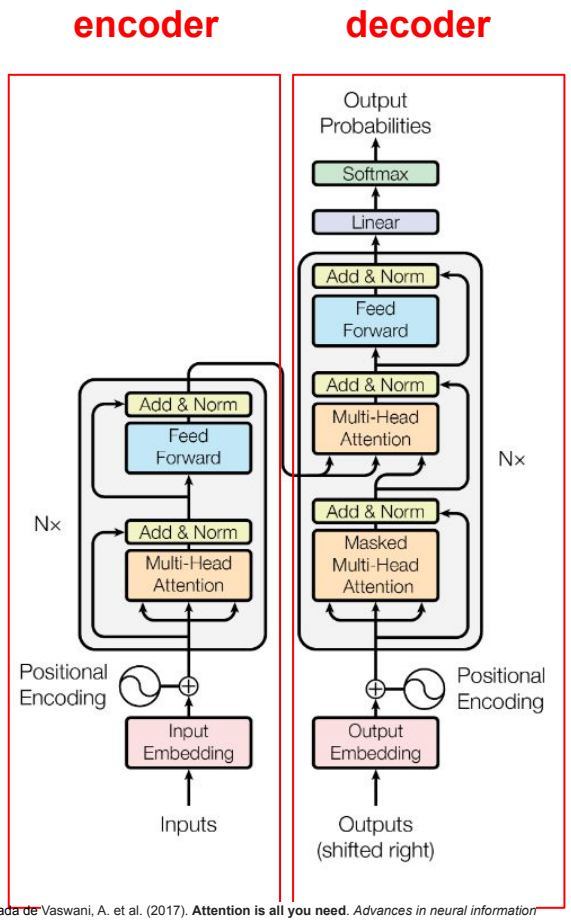


Figura tomada de Vaswani, A. et al. (2017). **Attention is all you need**. *Advances in neural information processing systems, 30.*

Primer modelo transformer

- El mecanismo de atención no permitía la paralelización.
- El mecanismo de **self-attention permite representar las palabras de la secuencia de entrada**, en función de la importancia con respecto al resto de palabras en la secuencia de entrada.
- En el **decodificador**, ocurre igual, pero el mecanismo enmascara las palabras siguientes de la secuencia de salida, y únicamente tiene en cuenta las palabras que ya se han producido.
- El mecanismo de **self-attention permite ser paralelizado** (multi-head).

Índice

- Seq2Seq
- Mecanismo de atención
- Primer Modelo transformer
- **Tipos de transformers**
- BERT
- Fine-tuning
- Conclusiones

Tipos de modelos transformadores

- Seq2Seq (traducción automática, generación de resúmenes)
- Modelos auto-encoding o codificadores(o encoders).
- Modelos auto-regresivos o decodificadores (decoders)

Modelos auto-encoding (encoders)

- Pre-entrenados utilizando la estrategia “Masked language modelling”.
- Esta estrategia consiste en enmascarar un % de los tokens de la secuencia de entrada. El modelo aprende a predecir dichos tokens.
- Su arquitectura coincide con el codificador de la arquitectura del transformer original.
- Este tipo de modelos se puede utilizar en muchas aplicaciones de PLN, pero su aplicación más natural es la clasificación de oraciones o la clasificación de tokens. BERT, ALBERT, Roberta o Distilbert son ejemplos de este tipo de modelos.

Modelos auto-regresivos (decoders)

- Pre-entrenados con tarea de predecir el próximo token, dada una secuencia de tokens previos.
- Su arquitectura coincide con el decodificador de la arquitectura del transformer original.
- Aunque estos modelos pueden ajustarse para muchas tareas de PLN, su aplicación más natural es la generación de texto. GPT es un ejemplo típico de estos modelos. Fue pre-entrenado sobre el conjunto de datos de Book Corpus. GPT-2 es una versión mejorada que fue pre-entrenado con el dataset WebText (páginas web de Reddit).

Índice

- Seq2Seq
- Mecanismo de atención
- Primer Modelo transformer
- Tipos de transformers
- **BERT**
- Fine-tuning
- Conclusiones

Qué es BERT

- BERT (Bidirectional Encoder Representations from Transformers) es un modelo transformer **auto-encoding** (es una pila de encoders).
- No tiene decoder.
- Es un **modelo de lenguaje contextualizado**. Es decir, el modelo no aprende un vector para cada palabra, sino para cada contexto donde ocurre una palabra.
- El contexto y el significado de una palabra están muy relacionados. *Distribución de Harris*: palabras que ocurren en el mismo contexto, suelen tener significados similares.

Qué es BERT

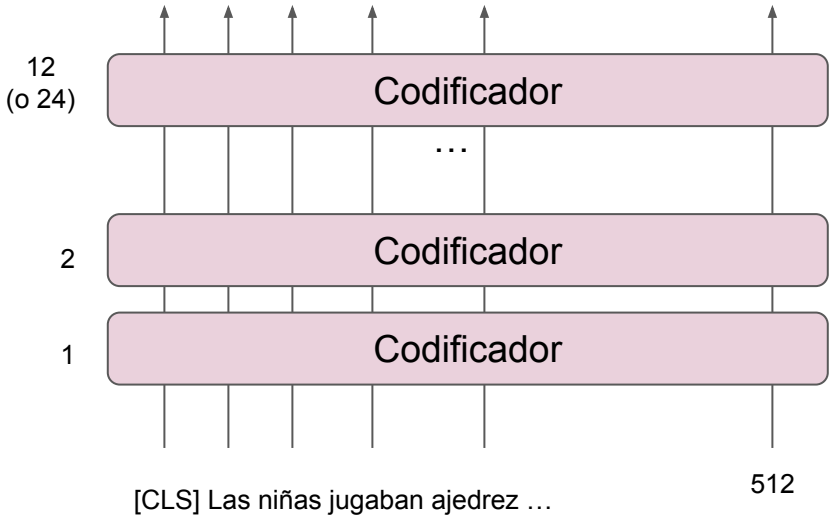
- BERT fue pre-entrenado utilizando de textos de Wikipedia (2,500 millones de tokens) y BookCorpus (más de 800 millones de tokens)
- Los vectores que proporciona BERT, aunque podrían ser directamente utilizados como entrada para distintas tareas de PLN, para obtener buenos resultados es necesario ajustar (**fine-tuning**) el modelo con un dataset específico de una tarea de NLP.

Qué es BERT

BERT es una pila de codificadores:

- BERT-base = 12 codificadores, tamaño del embedding= 768
- BERT-large = 24 codificadores, tamaño del embedding = 1024

El tamaño máximo de la secuencia de entrada siempre es 512



Estrategias para pre-entrenar BERT

- Se aplican a la vez:
  - Masked Language Model (MLM)
  - Next Sentence Prediction (NSP)

https://towardsdatascience.com/understanding-bert-bidirectional-encoder-representations-from-transformers-45ee6cd51eef

Masked Language Model (MLM)

- En cada oración de entrada se enmascaran el 15% de las palabras.
- BERT debe tratar de predecir dichos tokens a partir de sus contextos. Por ejemplo:
  - Entrada: *El <MASK1> no está hirviendo. El fuego debe estar <MASK2>.*
  - Labels: MASK1 = agua, MASK2= *apagado*

Next Sentence Prediction (NSP)

- El modelo recibe como entrada dos oraciones, y debe predecir si ocurren juntas o no.
- El conjunto de entrenamiento está formado por un 50% de pares oraciones que efectivamente son consecutivas, y el resto formado por pares de oraciones aleatorias.

- *Oración 1: El agua no está hirviendo.*
- *Oración 2: El fuego debe estar apagado.*
- *Label: IsNextSentence*

- *Oración 1: El agua no está hirviendo.*
- *Oración 2: Benzema es un futbolista francés.*
- *Label: NotNextSentence*

Modelos basados en BERT

- Dilistbert
  - No usa Next Sentence Prediction.
  - Sólo 6 encoders.
  - Más rápido, aunque con peores resultados que BERT.
- ALBERT
  - 12 capas, pero que comparten los parámetros.
  - Mejores resultados que BERT.
- ROBERTa
  - No usa Next Sentence Prediction.
  - Enmascara los tokens durante el entrenamiento.
- BETO (spanish), RigoBERTa
  - Entrenados con textos en Español.

Índice

- Seq2Seq
- Mecanismo de atención
- Primer Modelo transformer
- Tipos de transformers
- BERT
- **Fine-tuning**
- Conclusiones

Qué es fine-tuning?

- Consiste en ajustar (entrenar) un transformer pre-entrenado (por ejemplo, BERT) para una tarea concreta (por ejemplo, clasificación de textos) utilzando un dataset concreto para la tarea (por ejemplo, el dataset IMDB).
- Como el modelo pre-entrenado ya fue entrenado inicialmente sobre una gran colección de textos, la tarea de fine-tuning va a necesitar menos ejemplos (dataset) y tiempo para obtener resultados decentes.

Cómo ajustar un transformer pre-entrenado

- El transformer es extendido añadiendo nuevas capas necesarias para abordar la tarea.
- Por ejemplo, si la tarea es clasificación de textos, podríamos añadir una capa de softmax sobre la salida de BERT para predecir la clase para el texto de entrada.
- Si la tarea es NER, en lugar de un softmax, podríamos añadir un CRF que va a predecir la etiqueta correspondiente a cada token.
- El modelo extendido (modelo pre-entrenado y las nuevas capas) es entrenado sobre el conjunto de entrenamiento de la tarea que se está abordando.

Índice

- Seq2Seq
- Mecanismo de atención
- Primer Modelo transformer
- Tipos de transformers
- BERT
- Fine-tuning
- **Conclusiones**

Conclusiones

- Los modelos de lenguaje son imprescindibles en el desarrollo de sistemas de PLN.
- RNN no permite paralización y puede perder información en textos largos.
- Mecanismo de atención, y en particular, los transformers son una alternativa eficiente para RNN.
- BERT es un modelo de lenguaje basado en el contexto y capaz de representar palabras polisémicas.
- Cualquier modelo transformer puede ser fácilmente ajustado para realizar otras tareas de NLP.

OpenCourseWare
Procesamiento de Lenguaje Natural con Aprendizaje Profundo,

Gracias!!!

https://github.com/isegura