Introducción a procesamiento de lenguaje natural

CC5205 - Minería de Datos

Juan José Alegría

11 de noviembre de 2024

Basado en slides de Felipe Bravo (DCC) y Armand Joulin (Facebook)

Motivación

Entender (al menos un poco) cómo funciona ChatGPT

Agenda

- Pequeña introducción a NLP
- Modelos de lenguaje probabilísticos
- Modelos de lenguaje con redes neuronales
- Modelos de lenguaje grandes (LLMs)

Agenda

- **Pequeña introducción a NLP**
- Modelos de lenguaje probabilísticos
- Modelos de lenguaje con redes neuronales
- Modelos de lenguaje grandes (LLMs)

Procesamiento de lenguaje natural

“Natural Language Processing is the field of designing methods and algorithms that take as input or produce as output unstructured, natural language data”

(Neural Network Methods for Natural Language Processing, Goldberg, 2017)

Complejidades del lenguaje humano

- Ambigüedad:
  - Por ejemplo: “ayer comí pizza con amigos”, “ayer comí pizza con piña”, “ayer comí pizza con tenedor”. Las tres frases tienen la misma estructura, pero la palabra “con” modifica a distintas partes de la oración.
- Dinamismo: el lenguaje cambia y evoluciona con el tiempo
- Discreto: las letras son la unidad básica del lenguaje escrito, pero no se puede inferir la relación entre palabras a partir de sus letras. Ej: “guitarra” y “bajo”.
- Composicionalidad: el significado de una frase va más allá de las palabras que la conforman
- *Sparseness*: la cantidad de combinaciones que se pueden formar a partir de palabras (símbolos discretos) es prácticamente infinita => aun con un conjunto gigantesco de datos, es probable que encontremos frases que no aparecen en dicho conjunto

Agenda

- Pequeña introducción a NLP
- **Modelos de lenguaje probabilísticos**
- Modelos de lenguaje con redes neuronales
- Modelos de lenguaje grandes (LLMs)

Modelos de lenguaje

- Un **modelo de lenguaje** es una función que le asigna una probabilidad a cualquier oración
  - P(el perro ladra) = 0.001
  - P(ayer vi un capibara) = 0.0006
  - P(cuaderno gato guitarra estuche) = 0.000001

- Idea: un modelo de lenguaje asigna mayor probabilidad a frases con *sentido*, que sean *fluidas*

Modelos de lenguaje: ¿por qué?

- Reconocimiento de habla:
  - P(“wreak a nice beach”) < P(“recognize speech”)
  - P(“Vanilla, I scream”) < P(“vanilla ice scream”)
  - P(“voy a cocer un botón”) < P(“voy a coser un botón”)
- Optical character recognition (OCR)
  - P(“m0ve fast”) < P(“move fast”)

Modelos de lenguaje probabilísticos

- Modelaremos una oración *s* como una secuencia de palabras (tokens):

$$p(s) = p(w_1, w_2, \ldots, w_T)$$

- Esto lo podemos descomponer, usando probabilidades condicionales
- Para dos eventos *a* y *b*: $p(a, b) = p(b|a) * p(a)$
- Para tres eventos *a*, *b* y *c*:

$$p(a, b, c) = p(b, c|a) * p(a) = p(c|a, b) * p(b|a) * p(a)$$

- Aplicando recursivamente, cada token depende de los tokens previos

$$p(s) = p(w_1, w_2, \ldots, w_T) = \prod_{t=1}^{T} p(w_t|w_{t-1}, \ldots, w_1)$$

Modelos de lenguaje: estimación de parámetros

- Para estimar las probabilidades, necesitamos primero un conjunto de entrenamiento
  - En NLP, a los conjuntos de texto se les suele llamar *corpus*
- Teniendo nuestro corpus, podemos estimar las probabilidades de cada frase, simplemente contando
- Enfoque inicial: simplemente contar cuántas veces aparece cada oración, y dividir por el número de oraciones en el corpus de entrenamiento (N)

$$p(s) = p(w_1, w_2, \ldots, w_T) = \frac{count(w_1, w_2, \ldots, w_T)}{N}$$

Modelos de lenguaje: estimación de parámetros

- A medida que aumenta el largo de la oración, es menos probable que esté en el conjunto de entrenamiento.

  P(“el perro ladra”) >>>> P(“el perro ladra fuerte, porque es de noche y vio un gato que iba por caminando por el tejado, mientras los vecinos dormían y...”)

- Así, muchas frases largas tendrán probabilidad cero, lo cual hace que el modelo tenga baja generalización.

Modelos de lenguaje: estimación de parámetros

- Solución: acotar/limitar la memoria del modelo.
- En vez de que cada palabra dependa de todas las palabras previas, hacer que sólo depende de *n* tokens previos => Modelo de n-gramas.
- Modelo de unigramas:

$$p(s) = p(w_1, w_2, \ldots, w_T) = \prod_{t=1}^{T} p(w_t) = \prod_{t=1}^{T} \frac{count(w_t)}{N_{words}}$$

$$p(\text{el perro ladra}) = \frac{c(\text{el})}{N} * \frac{c(\text{perro})}{N} * \frac{c(\text{ladra})}{N}$$

Modelos de lenguaje: estimación de parámetros

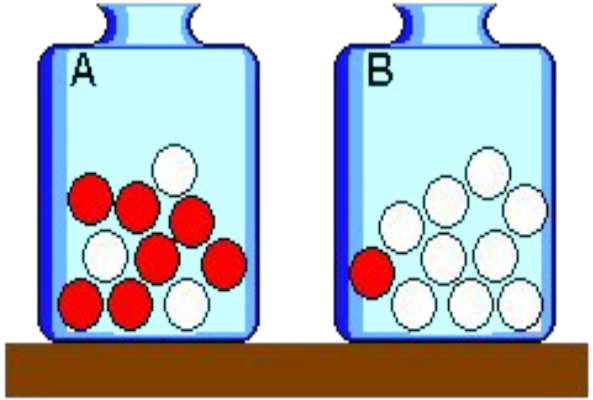
- Modelo de bigramas:

$$p(s) = p(w_1, w_2, \ldots, w_T) = \prod_{t=1}^{T} p(w_t|w_{t-1}) = \prod_{t=1}^{T} \frac{count(w_{t-1}, w_t)}{count(w_{t-1})}$$

$$p(\text{el perro ladra}) = \frac{c(\text{START, el})}{c(\text{START})} * \frac{c(\text{el perro})}{c(\text{el})} * \frac{c(\text{perro ladra})}{c(\text{perro})} * \frac{c(\text{ladra STOP})}{c(\text{ladra})}$$

Modelos de lenguaje generativos

- Los modelos de lenguaje pueden generar oraciones, haciendo un muestreo secuencial sobre las probabilidades
- Análogo a sacar bolitas de una urna, donde cada bolita tiene un tamaño proporcional a sus frecuencias relativas
- También sirve para predecir la palabra más probable; es decir, predecir la siguiente palabra



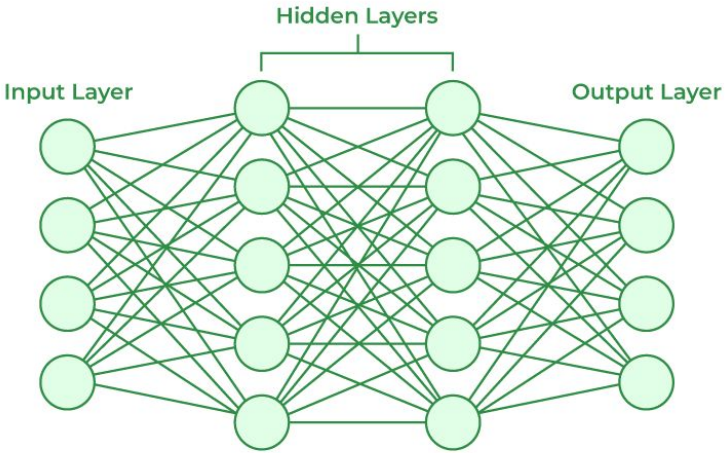
Problemas de modelos de lenguaje de n-gramas

- Dado que estamos truncando la ventana, no capturan contextos largos
  - *“Arturo Vidal es un futbolista chileno. Jugó en la liga alemana, española, italiana y brasileña. Su idioma nativo es…”*
- No capturan sinónimos ni dependencias entre palabras:
  - P(“vi un perro”) debiese ser similar a P(“observé un can”), pero el modelo no sabe que “perro” es lo mismo que “can”

Agenda

- Pequeña introducción a NLP
- Modelos de lenguaje probabilísticos
- **Modelos de lenguaje con redes neuronales**
- Modelos de lenguaje grandes (LLMs)

Redes neuronales



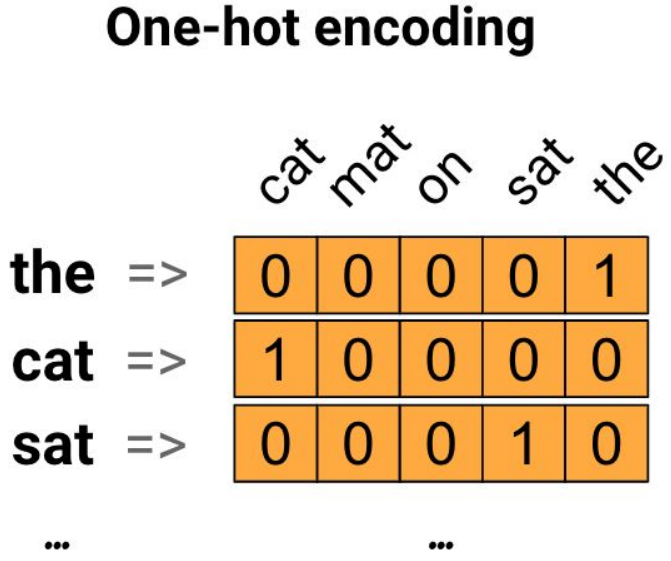
- Varias capas, cada una compuesta por *neuronas*
- En su versión más clásica, cada capa es una transformación lineal, sobre la cual se aplica una función no lineal
- En su conjunto, una red neuronal es una composición de funciones
- Se entrenan con *backpropagation*, un algoritmo para aplicar la regla de la cadena de derivadas de forma inteligente

Redes neuronales para texto

- Supongamos un problema de clasificación: dadas las palabras w_{t-2}, w_{t-1} , w_{t+1} y w_{t+2}, predecir la palabra central w_t.
- Clases posibles: todas las palabras del vocabulario, de tamaño $|V|$
- Primer inconveniente: ¿cómo representar cada palabra para ingresarla al modelo?

Vectores one-hot

- Cada palabra se representa como un vector de tamaño |V|. Este vector tiene ceros en todas las posiciones, excepto en la posición correspondiente a la palabra w_t. En dicha posición, tiene un 1.
- **Importante:** los vectores son ortogonales entre sí. Con codificación one-hot, no hay ninguna relación entre “libro” y “cuento”, o entre “silla” y “mesa”

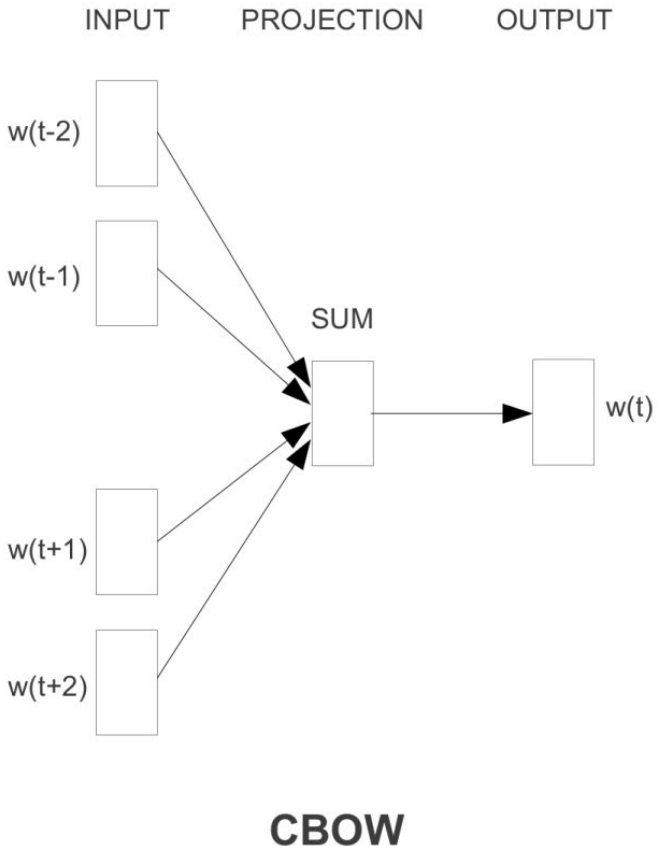


Redes neuronales para texto

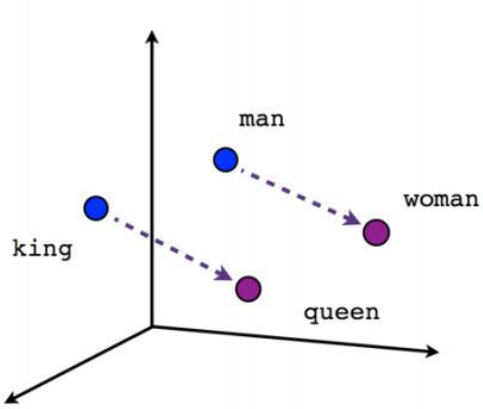
- Ahora ya sabemos cómo representar palabras para ingresarlas al modelo.
- Para la salida, esperamos un vector también de tamaño $|V|$, y queremos que la probabilidad sea muy alta en el índice de la palabra w_t.
  - Es decir, que prediga la clase (palabra) correcta
- Con esto, podemos entrenar una red para que, dadas palabras de contexto, prediga la palabra central
  - Esto es Word2Vec, con algoritmo Continuous Bag of Words (CBOW)

CBOW

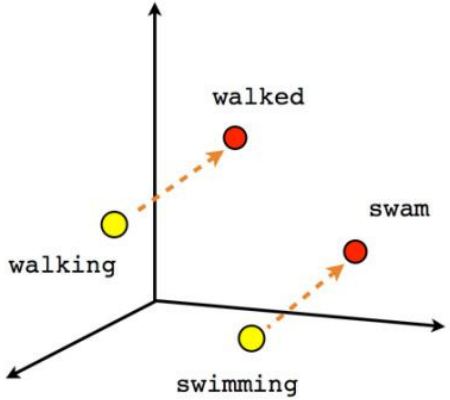
- Cada palabra es *proyectada* a una dimensión menor *d*
- La proyección se hace mediante una matriz *A* de tamaño |V| * *d*
- A cada palabra del vocabulario le corresponde una fila de la matriz *A*.
- Después de entrenar, se espera que cada fila sea una representación vectorial “buena” de la palabra correspondiente
- **Importante**: no nos importa la predicción de la red (los outputs los podemos descartar), sólo nos importa el aprendizaje de las representaciones internas (la matriz *A*).



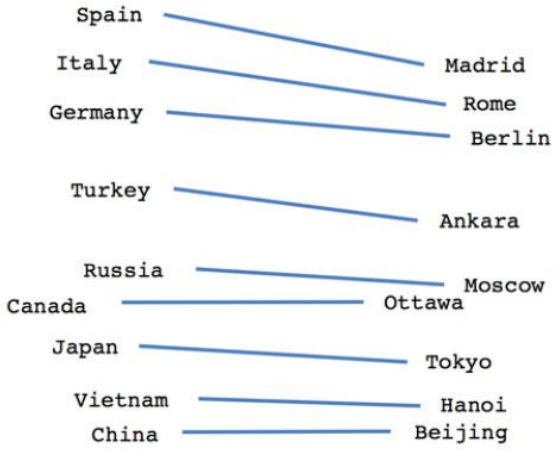
Word2Vec



Male-Female



Verb tense



Country-Capital

Word2Vec

- Se aprenden representaciones latentes (***embeddings***) para cada palabra, y estos vectores capturan relaciones semánticas entre distintos términos
- Esto permite realizar aritmética sobre las palabras (!!!)
- **Word2Vec** tiene además otro algoritmo, llamado Skip-gram: dada una palabra central, predecir palabras cercanas.
- Todo esto se basa en la hipótesis distribucional de lingüística [Harris, 1954]: palabras que ocurren en el mismo contexto tienden a tener significados similares.

Word2Vec

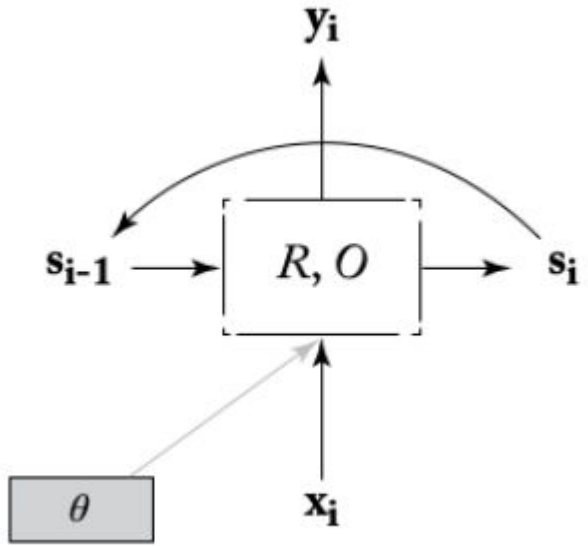
- Problema: ¿qué pasa con palabras con más de un significado (*polisemia*)?
  - “Me senté en el **banco** a darle comida a las palomas”
  - “Fui al **banco** a pedir un crédito”
- El modelo que acabamos de ver construye una *lookup table* para cada palabra. Así, la palabra **banco** tendrá el mismo embedding sin importar en qué contexto la estemos usando

Volviendo a modelos de lenguaje...

- ¿Cómo podemos modelar lenguaje usando redes neuronales?
- Formalmente: dada una secuencia de palabras w_1, w_2,..., w_t, queremos una distribución de probabilidades para la palabra w_{t+1}
- Primera opción: usar redes neuronales recurrentes (RNNs)

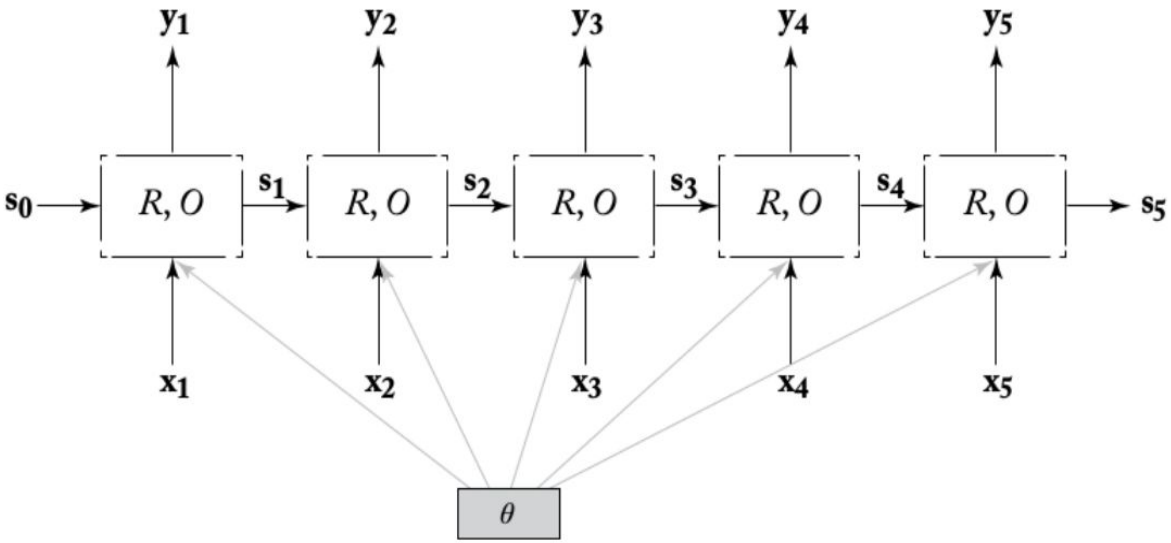
Redes neuronales recurrentes

- En cada paso, recibe un input x_i y un estado interno s_{i-1}
- Genera como salida un y_i, usando una función *O*
- Produce además un nuevo estado interno s_i usando la función *R*
- Está parametrizada por Θ **en todos los pasos**

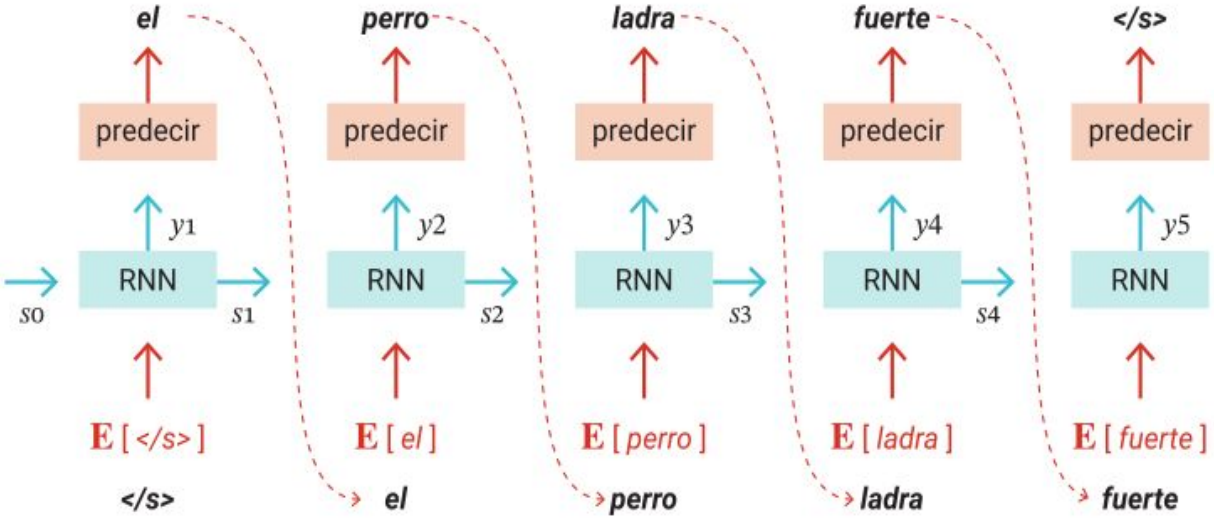


Redes neuronales recurrentes

- Si se “desenvuelve” la recursión, se ve algo así
- Se entrena con una adaptación de backpropagation (llamada *backpropagation through time*)



Redes neuronales recurrentes



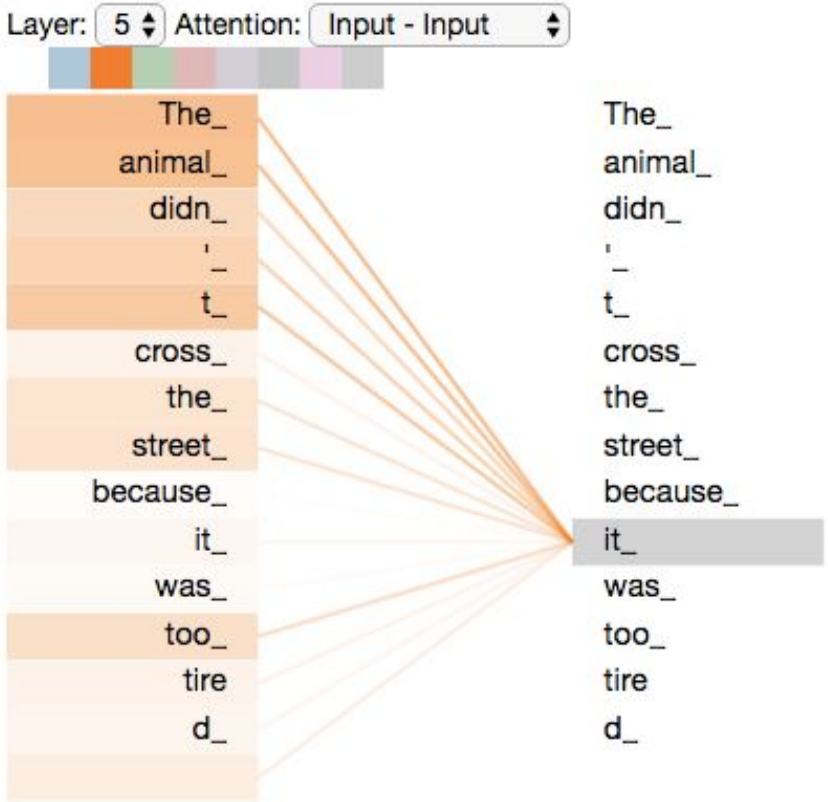
- Problema: *vanishing/exploding gradients*
- Varias arquitecturas proponen solucionarlo (GRU, LSTM)
- Otro problema: difícil de paralelizar

Agenda

- Pequeña introducción a NLP
- Modelos de lenguaje probabilísticos
- Modelos de lenguaje con redes neuronales
- **Modelos de lenguaje grandes (LLMs)**

Transformers

- Arquitectura reciente propuesta por Vaswani et al. (2017)
- Más paralelizable que RNNs
- Basada en un componente llamado **(auto)*atención***: una función que permite que el modelo decida qué partes del input son más importantes para generar una salida. Así, en cada paso el modelo tiene acceso a la oración completa y decide qué parte es más importante en el paso actual.



Transformers

- Paper original proponía una arquitectura basada en encoder-decoder: una red para codificar la entrada, y otra para decodificarla

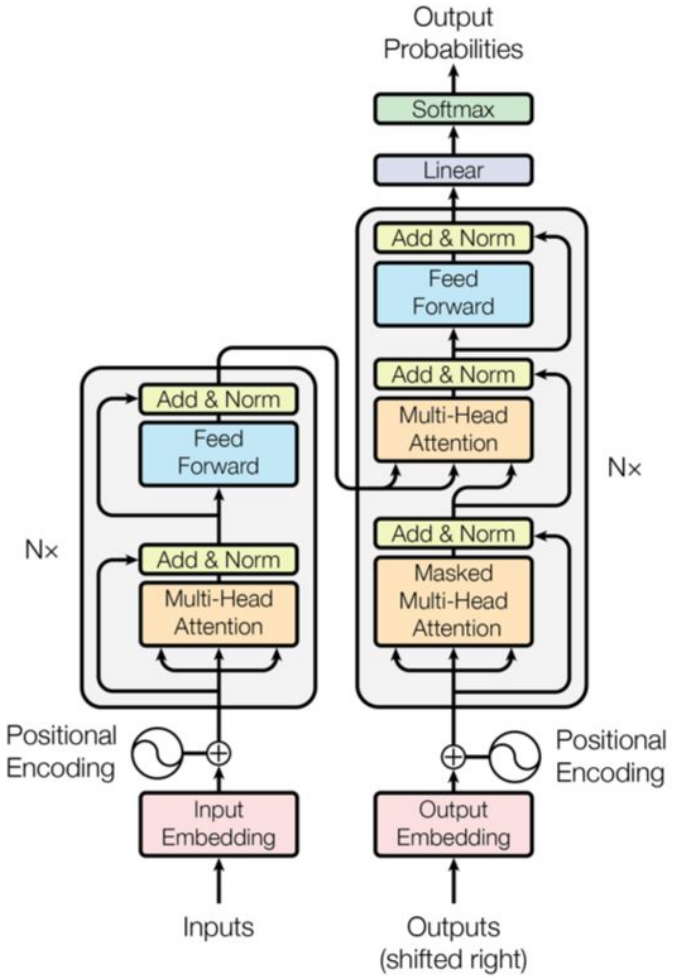


Figure 1: The Transformer - model architecture.

Transformers

- Encoder-only: enfocados en producir embeddings contextualizados (sin el problema de polisemia), y no sólo para palabras, sino para frases completas.
- En cada paso, pueden acceder a la oración completa.
- BERT, RoBERTa, BETO (versión en español y entrenada por profesores del DCC)

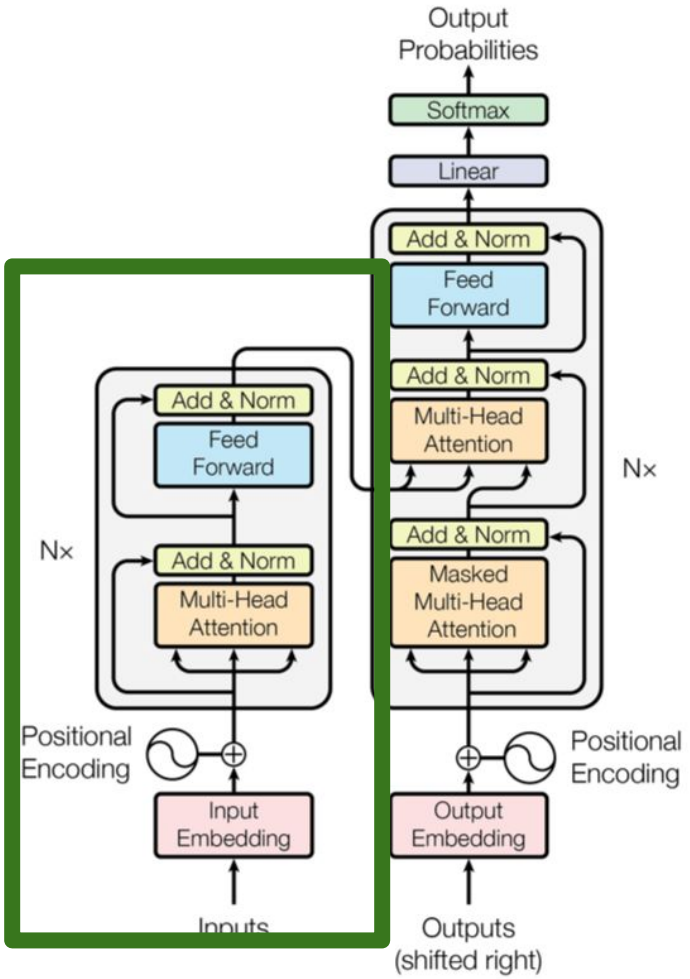


Figure 1: The Transformer - model architecture.

Transformers

- Decoder-only: enfocados en modelamiento de lenguaje y generación de texto.
- Sólo tienen acceso al texto previo, y tienen generación autorregresiva
- Ejemplos: GPT, Llama

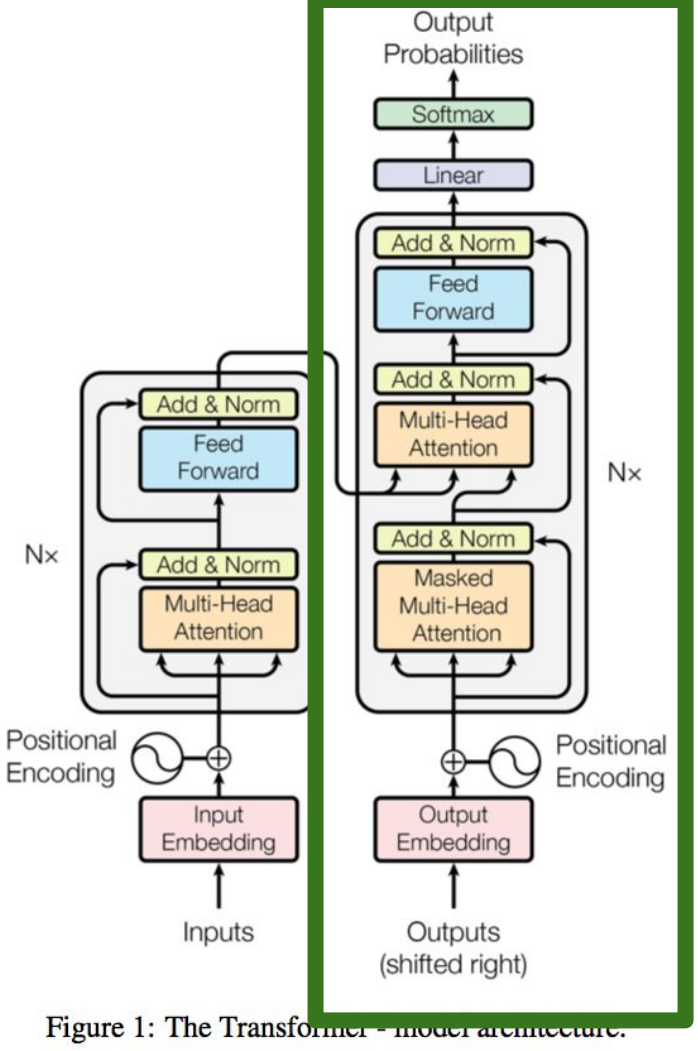


Figure 1: The Transformer - model architecture.

Generación autorregresiva

- En paso *t*, se tiene acceso a todos los tokens w_1, w_2, ...,w_{t-1}
- Se obtiene la distribución de probabilidades sobre los tokens, y se genera un token w_t
- Se añade el token w_t recién generado. Así, en paso t+1, se tiene acceso a w_1, w_2, ...,w_{t-1}, w_t

¿Cómo se entrenan estos modelos?

- En NLP más clásico, los modelos se entrenaban para una tarea *(task*) específica, con datos para esa tarea. Ejemplos:
  - Clasificación de sentimiento (“Me gustó esta película” => positivo, “odié este libro” => negativo)
  - Part of speech tagging (“El perro ladra” => ARTÍCULO SUSTANTIVO VERBO)
- Construir un dataset etiquetado es difícil/costoso
- Modelos nuevos se entrenan de manera *self-supervised*: se aprovecha la estructura del texto mismo.
- Se entrenan para predecir una palabra faltante en el medio de la oración, o palabras faltantes al final.

Modelos pre-entrenados

- Modelos como BERT, GPT, Llama, etc, fueron entrenados con cantidades gigantescas de datos
- Estos modelos ya tienen un conocimiento de lenguaje bastante profundo, lo cual se puede aprovechar para otras tareas (traducción de texto, etiquetado de partes de texto, clasificación de sentimiento, etc)
- Dos paradigmas:
  1. **Fine-tuning**: tomar la red ya entrenada, añadir una capa nueva encima de la red y re-entrenar, moviendo sólo un poquito los pesos ya aprendidos.
     - Paradigma clásico, muy usado en visión por computadora

Modelos pre-entrenados

2. **In-context learning:** el mismo paper que presenta GPT 3, mostró que los modelos de lenguaje grandes tienen la capacidad de aprender vía ejemplos en el prompt, *sin necesidad de reentrenar*

Zero-shot

The model predicts the answer given only a natural language description of the task. No gradient updates are performed.

```
1  Translate English to French:        ← task description
2  cheese =>                           ← prompt
```

One-shot

In addition to the task description, the model sees a single example of the task. No gradient updates are performed.

```
1  Translate English to French:        ← task description
2  sea otter => loutre de mer          ← example
3  cheese =>                           ← prompt
```

Few-shot

In addition to the task description, the model sees a few examples of the task. No gradient updates are performed.

```
1  Translate English to French:        ← task description
2  sea otter => loutre de mer          ← examples
3  peppermint => menthe poivrée        ←
4  plush girafe => girafe peluche      ←
5  cheese =>                           ← prompt
```

Entonces... ¿qué es un LLM?

“Typically, *large language models* (LLMs) refer to Transformer language models that contain hundreds of billions (or more) of parameters, which are trained on massive text data” (A Survey of Large Language Models, Zhao, 2024)

¿Y qué es GPT?

- Es un *large language model*
- GPT: Generative Pre-trained Transformer
  - *Generative*: puede generar palabras, de la misma forma que un modelo de lenguaje normal
  - *Pre-trained*: pre-entrenado en conjuntos gigantescos de datos, de tal forma que modela muy bien el lenguaje
  - *Transformer*: arquitectura de redes neuronales empleada en el modelo
- Funciona de manera autorregresiva

Cosas que no vimos

- Pre-procesamiento de texto (eliminar stopwords, lematización, stemming)
- Modelos clásicos para clasificar documentos (Bag of Words + TF-IDF)
- Distintas tareas de NLP: clasificación de texto, NER, machine translation, summarization, etc
- Técnicas de prompting (chain-of-thought)
- Propiedades emergentes de LLMs
- ¿Cuál es la diferencia entre GPT y ChatGPT?
- Y mucho más...

“““Conclusiones”””””

- Trabajar con texto es complejo
- Hay mucha, mucha, muuuuucha investigación ocurriendo en esta área
- La investigación está avanzando muy rápido