Grau en Intel·ligència Artificial

Procesamiento de Lenguaje Humano
12. Modelos de Lenguaje

Índice

Outline

Modelado del Lenguaje

El Modelado del Lenguaje es la tarea de predecir qué palabra sigue a otra.

- De manera más formal: dada una secuencia de palabras, calcular la distribución de probabilidad de la siguiente palabra:

$$P(w_t|w_1, w_2, \ldots, w_{t-1})$$

  donde w_t puede ser cualquier palabra del vocabulario.
- Un sistema que realiza esto se llama un Modelo de Lenguaje.

Ej: The students opened their w_n

- $w_n^1 =$ books
- $w_n^2 =$ laptops
- $w_n^3 =$ exams

Usas ML Todos los Días (I)

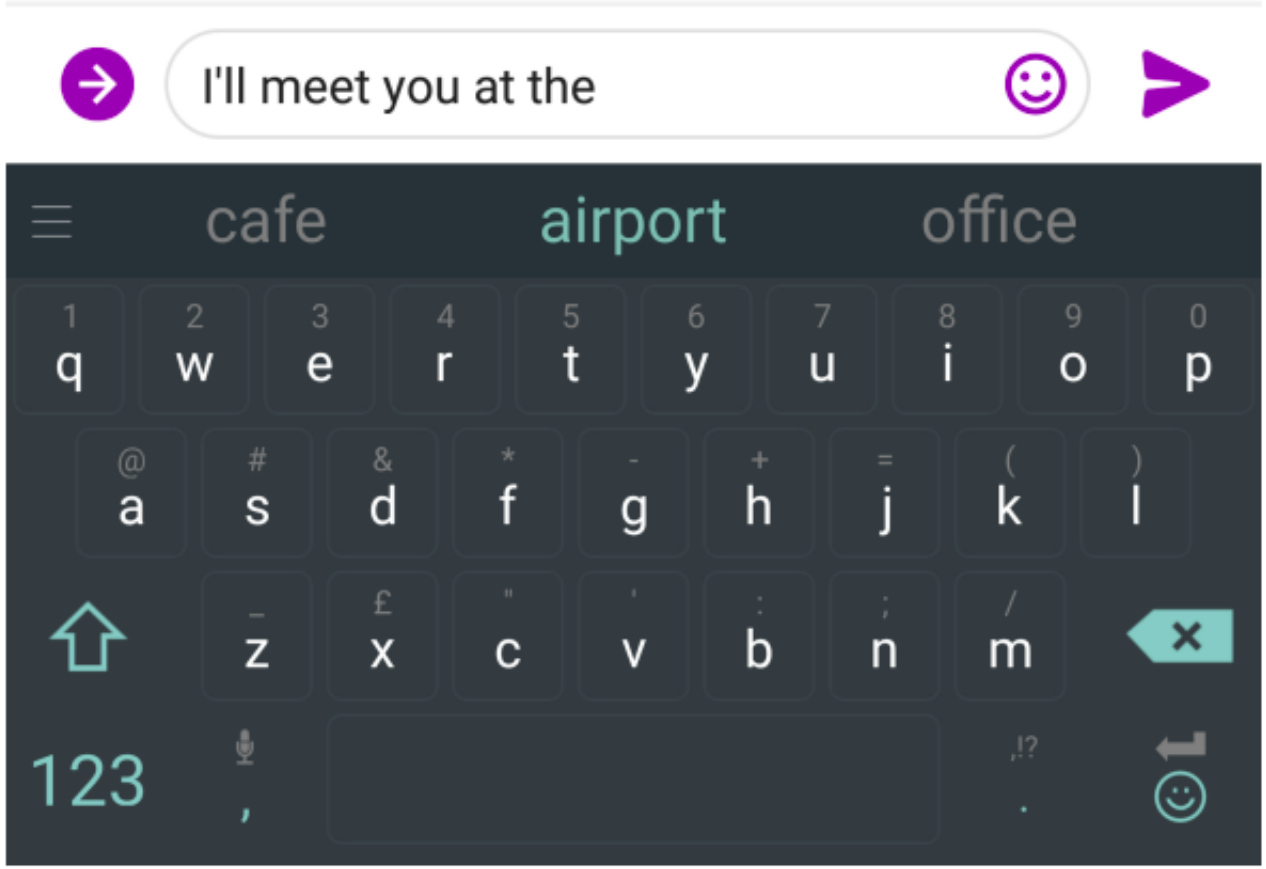


Figure: Ejemplo de ML: Predicción en el teclado de Google

Usas ML Todos los Días (II)

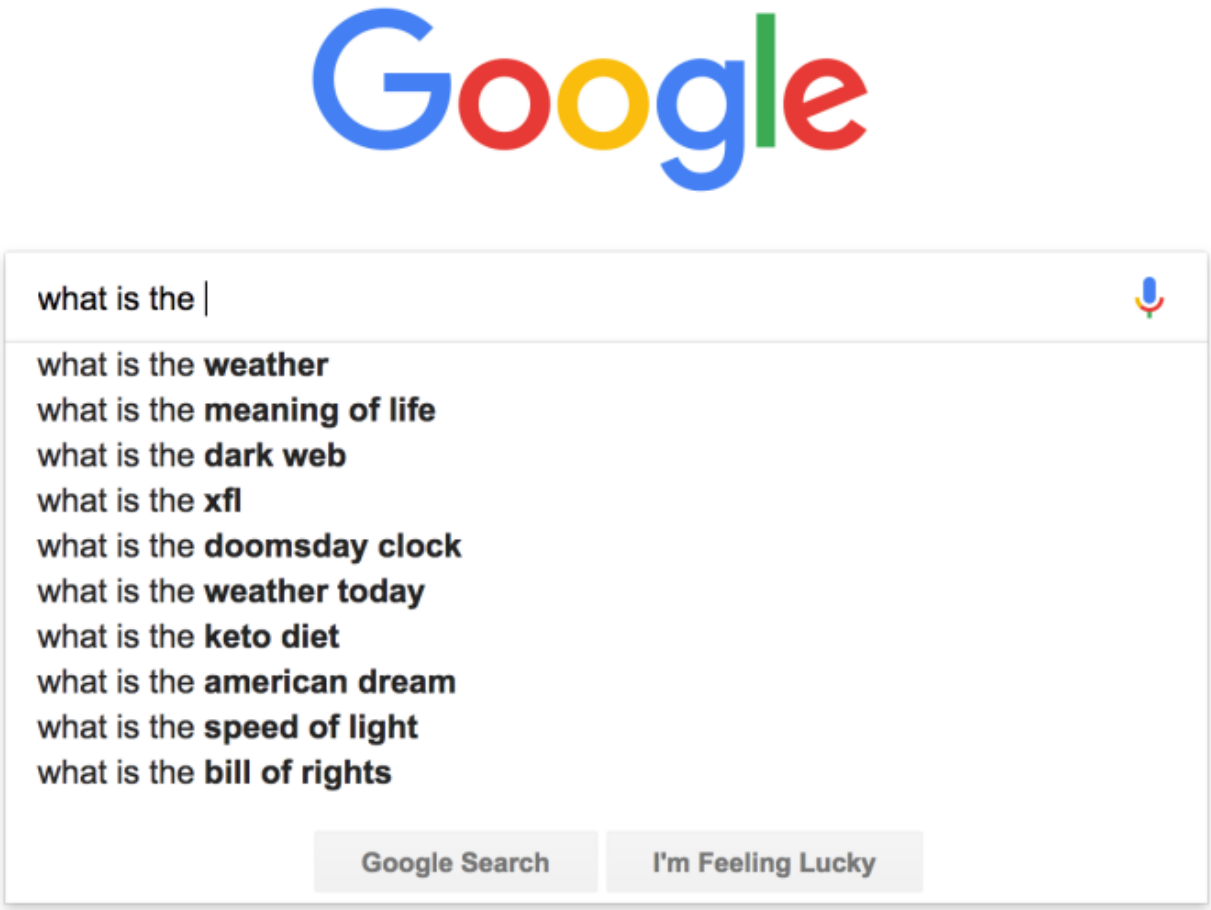


Figure: Ejemplo de ML: Sugerencias en la búsqueda de Google

Modelos de Markov para Modelado del Lenguaje

- **Modelado del Lenguaje:** predecir la probabilidad de una secuencia de palabras $W = w_1, w_2, ..., w_T$ dada la context $C = w_{1-k}, ..., w_{t-1}$.
- **Recuerda**: Podemos usar Modelos de Markov (MMs) para modelar **secuencias de palabras** - o modelar lenguaje.
- Propuesta: **MM de bigrama**. Es decir, utilizar Modelos de Markov para modelar la secuencia de palabras como una secuencia de estados, donde cada estado representa una palabra en particular y cada arista representa la probabilidad de transición entre dos palabras.

Modelos de Markov para Modelado del Lenguaje (II)

- Fórmulas de los MM:
  - Probabilidades de transición de estados: $a_{i,j} = P(q_t = j | q_{t-1} = i)$
  - Probabilidades de emisión: $b_i(w_t) = P(w_t | q_t = i)$
  - Probabilidades iniciales de los estados: $\pi_i = P(q_1 = i)$
- Aplicado a un Modelo de Lenguaje:
  - Probabilidad conjunta de palabras y estados:

$$P(W, Q) = \pi_{q_1} b_{q_1}(w_1) \prod_{t=2}^{T} a_{q_{t-1}, q_t} b_{q_t}(w_t)$$

  - Probabilidad de palabras dado el contexto:

$$P(W|C) = \sum_{q_1, \ldots, q_T} P(W, Q) = \sum_{q_1} \pi_{q_1} b_{q_1}(w_1) \sum_{q_2} a_{q_1, q_2} b_{q_2}(w_2$$

  - Probabilidad de la próxima palabra dadas las palabras anteriores:

$$P(w_t | w_{1:t-1}) \propto \sum_{q_t} a_{q_{t-1}, q_t} b_{q_t}(w_t)$$

Modelos de Lenguaje N-grama

- El Modelado del Lenguaje predice la distribución de probabilidad de la siguiente palabra en una secuencia dada de palabras.
- Antes del aprendizaje profundo: **Modelo de Lenguaje n-grama**
  - Un n-grama es un conjunto de n palabras consecutivas.
  - Ejemplos de n-gramas:
    1. Unigrama: "the", "students", "opened", "their"
    2. Bigrama: "the students", "students opened"
    3. Trigrama: "the students opened", "students opened their"
    4. 4-grama: "the students opened their"
  - Recopilar estadísticas sobre la frecuencia de diferentes n-gramas y utilizarlas para predecir la próxima palabra.
- Deep Learning: Redes Neuronales Recurrentes, Transformers...

Modelos de Lenguaje N-grama (II)

- Markov Assumption: el Modelado del Lenguaje depende solo de las n-1 palabras anteriores.
- Podemos calcular las probabilidades de n-gramas y (n-1)-gramas contándolos en un corpus grande.
- En el Modelado del Lenguaje, queremos predecir la distribución de probabilidad de la siguiente palabra dadas las palabras anteriores.
- La probabilidad condicional se define como:

$$P(w_n|w_{n-1}, w_{n-2}, ..., w_1) = \frac{P(w_n, w_{n-1}, w_{n-2}, ..., w_1)}{P(w_{n-1}, w_{n-2}, ..., w_1)}$$

- Podemos utilizar Modelos de Lenguaje n-grama para estimar la probabilidad condicional de la próxima palabra dadas las palabras precedentes.

Modelos de Lenguaje N-grama (III)

$$P(\boldsymbol{x}^{(t+1)}|\boldsymbol{x}^{(t)},\ldots,\boldsymbol{x}^{(1)}) = P(\boldsymbol{x}^{(t+1)}|\overbrace{\boldsymbol{x}^{(t)},\ldots,\boldsymbol{x}^{(t-n+2)}}^{n\text{-1 words}})$$

(assumption)

Figure: Markov Assumption para modelos n-grama

prob of a n-gram

prob of a (n-1)-gram

$$= \frac{P(\boldsymbol{x}^{(t+1)},\boldsymbol{x}^{(t)},\ldots,\boldsymbol{x}^{(t-n+2)})}{P(\boldsymbol{x}^{(t)},\ldots,\boldsymbol{x}^{(t-n+2)})}$$

Figure: Componentes de $P(w_n|w_{n-1},w_{n-2},...,w_1)$

Modelos de Lenguaje N-grama (IV)

~~as the proctor started the clock, the~~ students opened their _____

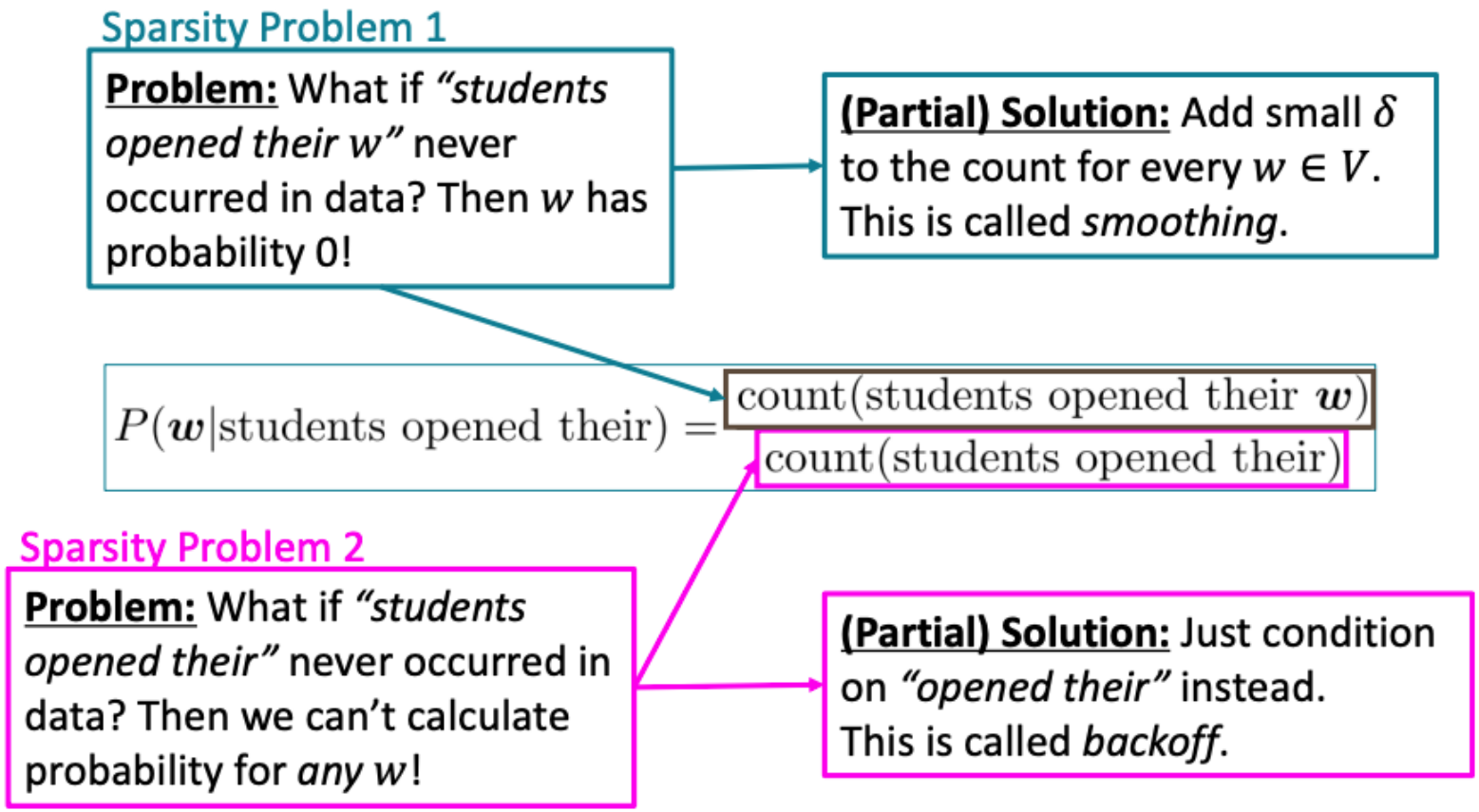
discard

condition on this

$$P(\boldsymbol{w}|\text{students opened their}) = \frac{\text{count}(\text{students opened their } \boldsymbol{w})}{\text{count}(\text{students opened their})}$$

- Por ejemplo, supongamos que en el corpus:
  1. "students opened their" ocurrió 1000 veces.
  2. "students opened their books" ocurrió 400 veces: $p(books|students\_opened\_their) = 0.4$
  3. "students opened their exams" ocurrió 100 veces: $p(exams|students\_opened\_their) = 0.1$

Problema de Sparsity en n-gram LM



Problema de Sparsity en n-gram LM (II)

- **Recuerda**, solución 1 - **Suavizado con adición de k**: agregar una pequeña constante k al recuento de cada n-grama.

$$P_{\text{add-k}}(w_n|w_{n-1},\ldots,w_{n-k+1}) = \frac{c(w_{n-k+1},\ldots,w_n)+k}{c(w_{n-k+1},\ldots,w_{n-1})+Vk}$$

  - $c(w_{n-k+1},\ldots,w_n)$ es el recuento del n-grama $w_{n-k+1},\ldots,w_n$ en el corpus.
  - $c(w_{n-k+1},\ldots,w_{n-1})$ es el recuento del (n-1)-grama $w_{n-k+1},\ldots,w_{n-1}$ en el corpus.
  - V es el tamaño del vocabulario.
  - k es el parámetro de suavizado.
  - Por lo tanto, la probabilidad de un n-grama con un recuento de cero en el corpus nunca es cero, y la suma de las probabilidades de todos los posibles n-gramas siempre es igual a 1.

Problema de Sparsity en n-gram LM (III)

- Solución 2 - **retroceso (backoff)**: estimar recursivamente la probabilidad de un n-grama utilizando n-gramas de orden inferior cuando el recuento del n-grama de orden superior es cero o muy pequeño.

$$P_{\text{bo}}(w_n|w_{n-1},\ldots,w_{n-k+1}) =$$

$$\begin{cases} \alpha_{w_{n-k+1},\ldots,w_{n-1}} P(w_n|w_{n-1},\ldots,w_{n-k+2}) & \text{si } c(w_{n-k+1},\ldots,w_n)>0 \\ \beta_{w_{n-k+2},\ldots,w_{n-1}} P_{\text{bo}}(w_n|w_{n-1},\ldots,w_{n-k+2}) & \text{en otro caso} \end{cases}$$

Donde:

- $\alpha_{w_{n-k+1},\ldots,w_{n-1}}$ y $\beta_{w_{n-k+2},\ldots,w_{n-1}}$ son constantes de normalización que aseguran que las probabilidades sumen 1.
- $P(w_n|w_{n-1},\ldots,w_{n-k+2})$ es la probabilidad del (n-1)-grama $w_{n-k+2},\ldots,w_{n-1}$.

Problema de Sparsity en n-gram LM (IV)

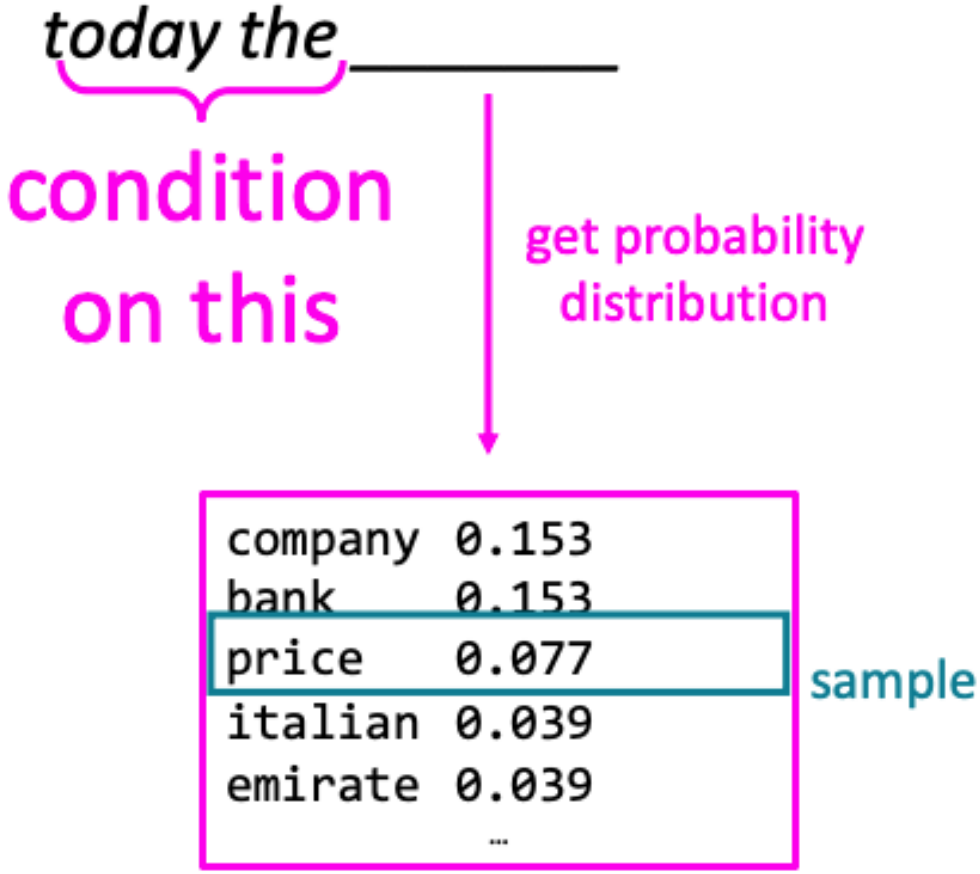
Storage: Need to store count for all *n*-grams you saw in the corpus.

$$P(\boldsymbol{w}|\text{students opened their}) = \frac{\text{count}(\text{students opened their } \boldsymbol{w})}{\text{count}(\text{students opened their})}$$

Outline

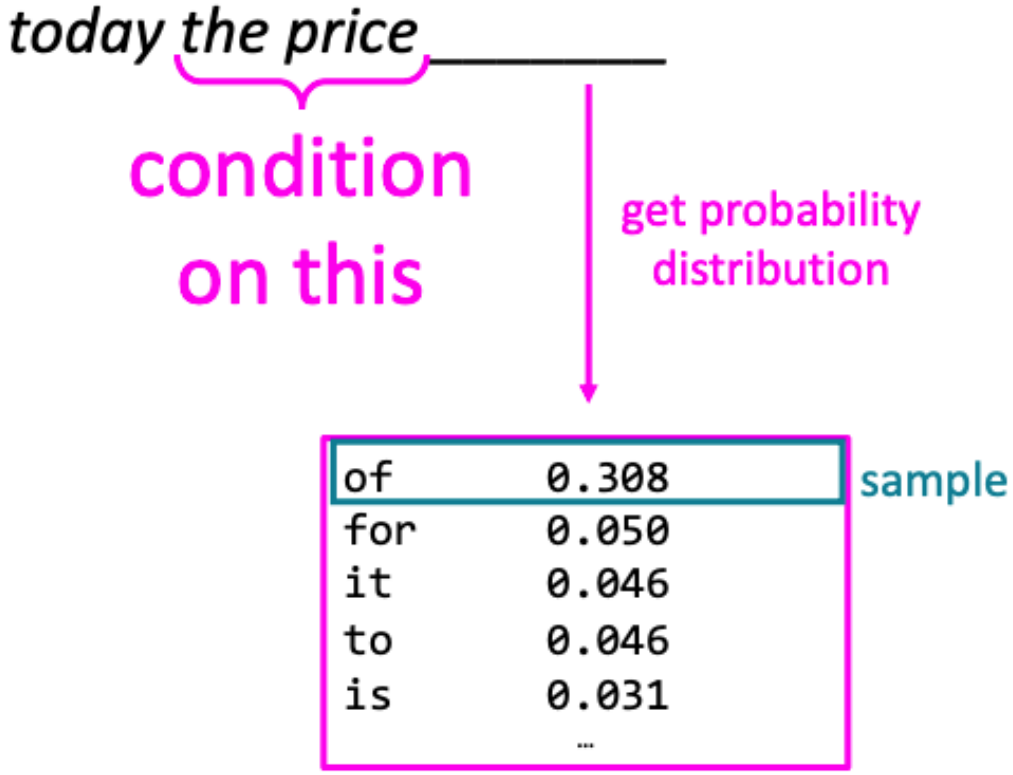
Generación con Modelos de Lenguaje n-grama

También puedes utilizar los Modelos de Lenguaje para generar texto:



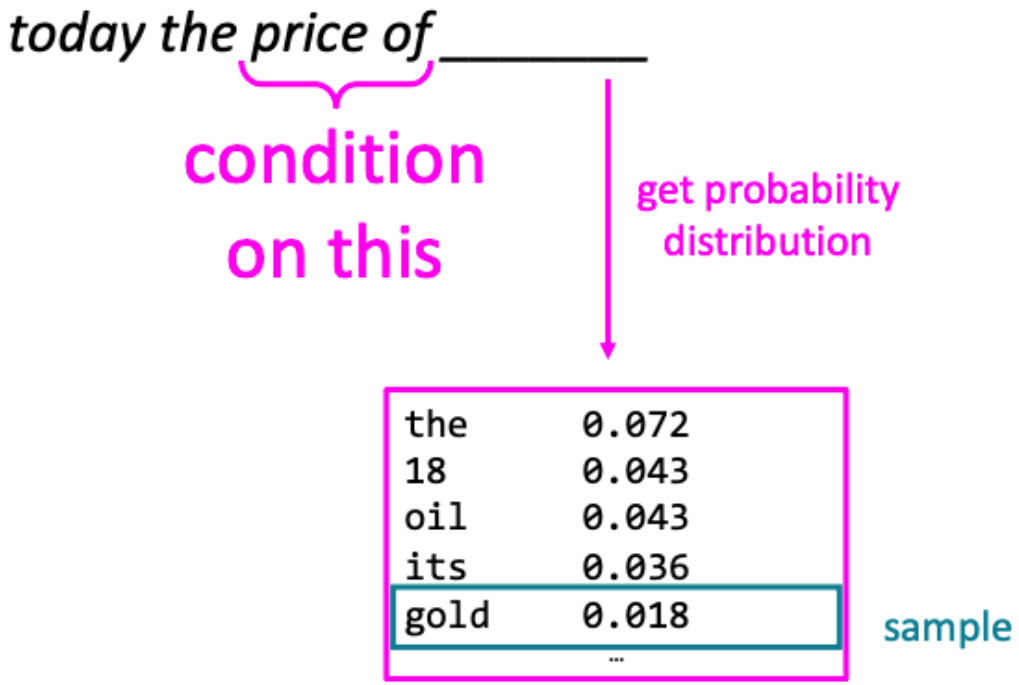
Generación con Modelos de Lenguaje n-grama (II)

También puedes utilizar los Modelos de Lenguaje para generar texto:



Generación con Modelos de Lenguaje n-grama (III)

También puedes utilizar los Modelos de Lenguaje para generar texto:



Generación con Modelos de Lenguaje n-grama (IV)

También puedes utilizar los Modelos de Lenguaje para generar texto:

> *Today the price of gold per ton, while production of shoe lasts and the shoe industry, the bank intervened just after it considered and rejected an IMF demand to rebuild depleted European stocks, sept 30 end primary 76 c a share.*

- ¡Gramaticalmente correcto!
- Pero incoherente. Necesitamos considerar más de 3 palabras a la vez si queremos modelar el lenguaje de manera adecuada. Pero aumentar n empeora el problema de sparsity y aumenta el tamaño del modelo.

Outline

Modelo de Lenguaje Neuronal

- Recordemos la tarea de Modelado de Lenguaje:
  - Entrada: secuencia de palabras: $w_1, w_2, ..., w_n$
  - Salida: distribución de probabilidad de las siguientes palabras: $P(w_{n+1}|w_1, w_2, ..., w_n)$
- ¿Qué tal un modelo neural basado en ventanas?
  - Vimos esto aplicado a Reconocimiento de Entidades Nombradas:

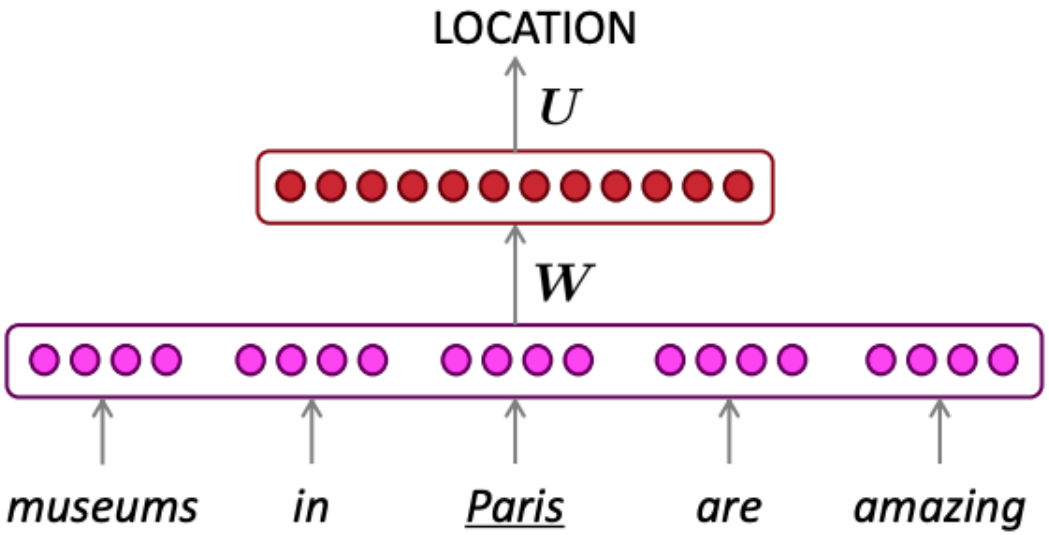


Figure: Modelo neural basado en ventanas para Reconocimiento de Entidades Nombradas

Modelo de Lenguaje Neuronal (II)

output distribution

$$\hat{\boldsymbol{y}} = \text{softmax}(\boldsymbol{U}\boldsymbol{h} + \boldsymbol{b}_2) \in \mathbb{R}^{|V|}$$

hidden layer

$$\boldsymbol{h} = f(\boldsymbol{W}\boldsymbol{e} + \boldsymbol{b}_1)$$

concatenated word embeddings

$$\boldsymbol{e} = [\boldsymbol{e}^{(1)}; \boldsymbol{e}^{(2)}; \boldsymbol{e}^{(3)}; \boldsymbol{e}^{(4)}]$$

words / one-hot vectors

$$\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \boldsymbol{x}^{(3)}, \boldsymbol{x}^{(4)}$$

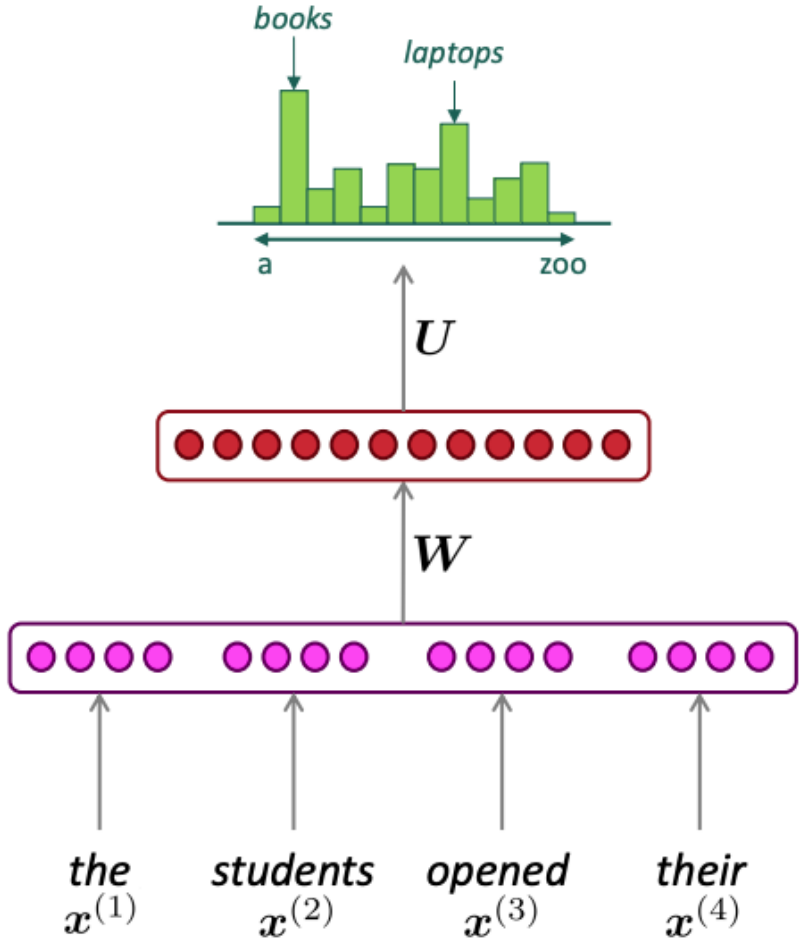


Figure: Modelo de ventana fija para el modelo de lenguaje. Aproximadamente: Y. Bengio, et al. (2000/2003): A Neural Probabilistic Language Model

Modelo de Lenguaje Neuronal (III)

- Mejoras sobre el modelo n-grama:
  - No hay problema de sparsity
  - No es necesario almacenar todos los n-gramas observados
- Problemas que persisten:
  - La ventana fija es demasiado pequeña
  - Aumentar la ventana aumenta W
  - Cada word vector se multiplica por diferentes pesos en W.

Redes Neuronales Recurrentes

Necesitamos una arquitectura neural que pueda procesar entradas de cualquier longitud.

- Modelo de Lenguaje n-grama:
  $P(w_n|w_{n-1}, w_{n-2}, ..., w_{n-N+1})$
- Modelo neuronal basado en ventanas:
  $P(w_{n+1}|w_n, w_{n-1}, ..., w_{n-m+1})$
- Red Neuronal Recurrente: $\mathbf{h}_t = f(\mathbf{x}_t, \mathbf{h}_{t-1})$

Outline

RNN para el Modelado del Lenguaje

- Los modelos de lenguaje basados en RNN son un tipo de modelo de lenguaje que utiliza RNNs para predecir la siguiente palabra en una secuencia.
- **Idea principal:** Aplicar repetidamente los mismos pesos W.

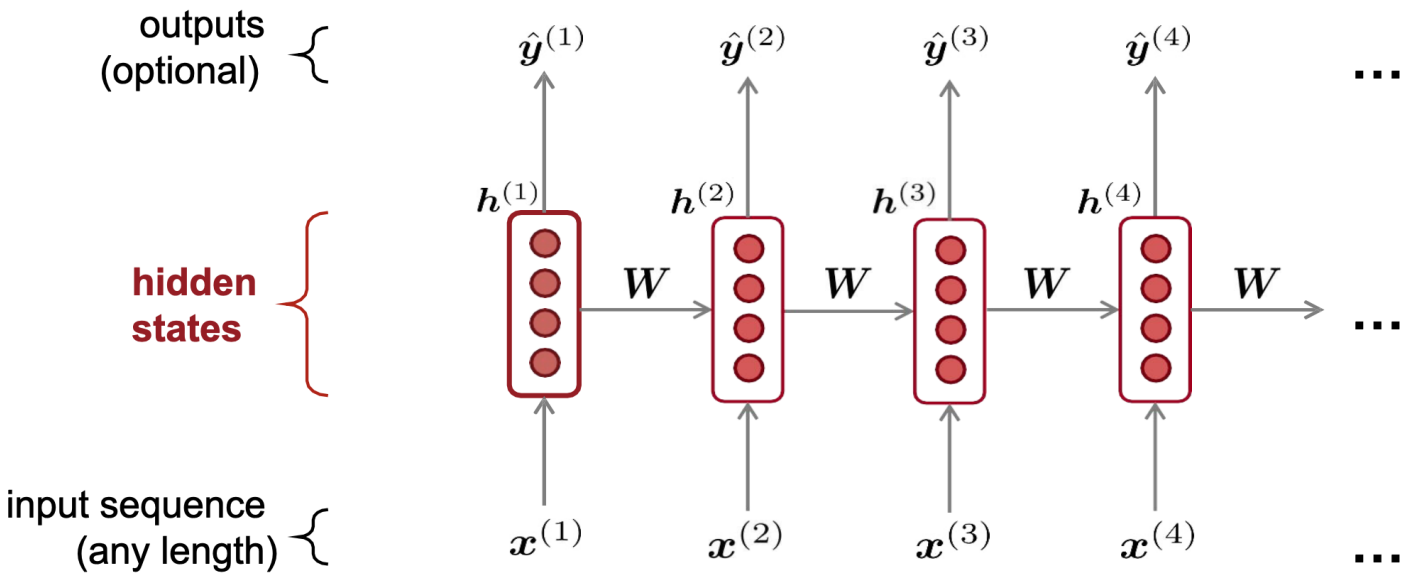


Figure: Representación de una RNN básica

Conceptos básicos de las RNN

Recuerda que:

- Las RNN están diseñadas para modelar datos secuenciales procesando un elemento de la secuencia a la vez, mientras mantienen un estado interno.
- En cada paso de tiempo t, la RNN recibe un vector de entrada x_t y un vector de estado oculto h_{t-1}, y produce un vector de salida y_t y un nuevo vector de estado oculto h_t.
- El vector de estado oculto sirve como memoria de los elementos anteriores en la secuencia, permitiendo que la RNN capture dependencias a largo plazo.

Modelado del Lenguaje con RNN

- En el modelado del lenguaje con RNN, la secuencia de entrada consiste en palabras $x_1, x_2, \ldots, w_n$.
- En cada paso de tiempo, la RNN recibe la palabra actual x_t y el vector de estado oculto anterior h_{t-1}, y produce una distribución de probabilidad sobre el vocabulario de posibles siguientes palabras.
- El vector de salida y_t es una distribución de probabilidad sobre el vocabulario.
- La RNN se entrena para minimizar el logaritmo negativo de la probabilidad verdadera de la siguiente palabra dada las palabras anteriores en la secuencia.

$$y_t = \text{softmax}(W_{hy} h_t + b_y)$$

$$\mathcal{L} = -\sum_{t=1}^{T} \log y_{t,\text{verdadero}}$$

Modelado del Lenguaje con RNN (II)

output distribution

$$\hat{\boldsymbol{y}}^{(t)} = \text{softmax}\left(\boldsymbol{U}\boldsymbol{h}^{(t)} + \boldsymbol{b}_2\right) \in \mathbb{R}^{|V|}$$

hidden states

$$\boldsymbol{h}^{(t)} = \sigma\left(\boldsymbol{W}_h\boldsymbol{h}^{(t-1)} + \boldsymbol{W}_e\boldsymbol{e}^{(t)} + \boldsymbol{b}_1\right)$$

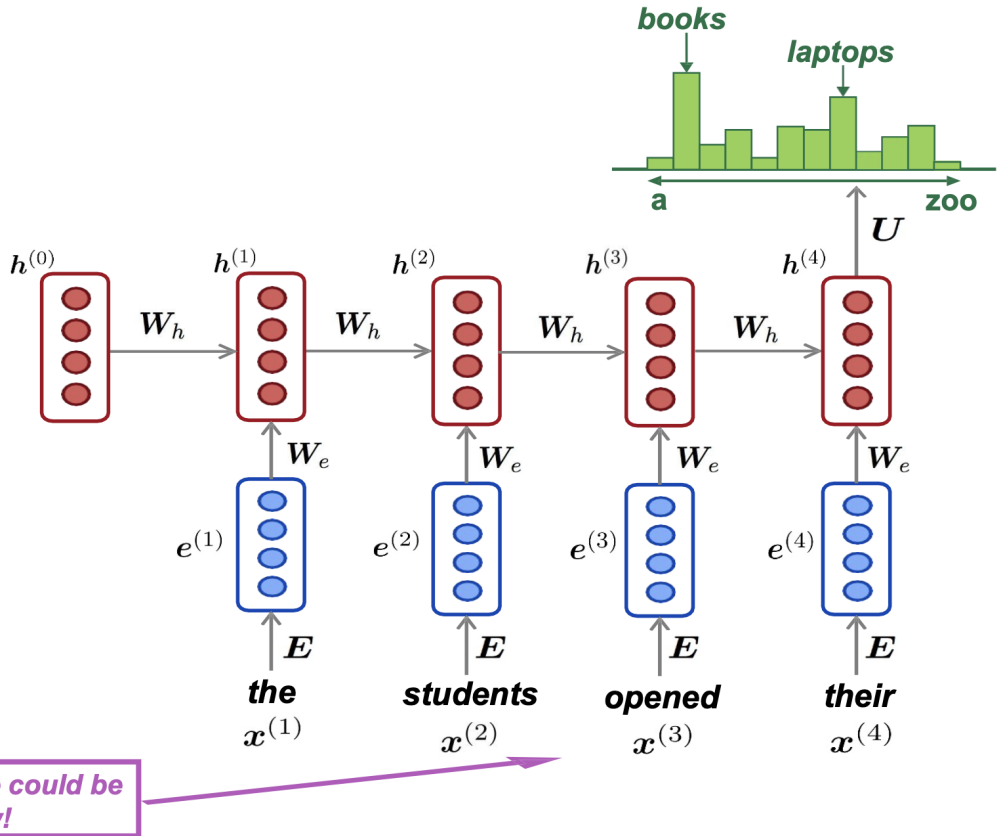
$\boldsymbol{h}^{(0)}$ is the initial hidden state

word embeddings

$$\boldsymbol{e}^{(t)} = \boldsymbol{E}\boldsymbol{x}^{(t)}$$

words / one-hot vectors

$$\boldsymbol{x}^{(t)} \in \mathbb{R}^{|V|}$$



Note: this input sequence could be much longer now!

Modelado del Lenguaje con RNN (III)

Ventajas de las RNN:

- Pueden procesar entradas de cualquier longitud.
- El cálculo para el paso t puede (en teoría) utilizar información de muchos pasos anteriores.
- El
  tamaño del modelo no aumenta para entradas más largas.
- Los mismos pesos se aplican en cada paso de tiempo, por lo que hay simetría en cómo se procesan las entradas.

Desventajas de las RNN:

- El cálculo recurrente es lento.
- En la práctica, es difícil acceder a información de muchos pasos anteriores.

Outline

Entrenamiento de un Modelo de Lenguaje RNN

- Obtener un corpus grande de texto que sea una secuencia de palabras.
- Alimentar al RNN-LM; calcular la distribución de salida para cada paso t.
  - Es decir, predecir la distribución de probabilidad de cada palabra, dadas las palabras hasta el momento.
- La función de loss en el paso t es la entropía cruzada entre la distribución de probabilidad predicha P_t y la verdadera próxima palabra w_{t+1} (one-hot para w_{t+1}):

$$\mathcal{L}_t = -\sum_{i=1}^{|V|} 1_{\{w_{t+1}=i\}} \log P_t(i|w_1, \ldots, w_t)$$

- Calcular el promedio para obtener la loss general para todo el conjunto de entrenamiento:

$$\mathcal{L} = \frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T_i}\mathcal{L}_t = -\frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T_i}\sum_{j=1}^{|V|} 1_{\{w_{t+1}=j\}} \log P_t(j|w_1, \ldots, w_t)$$

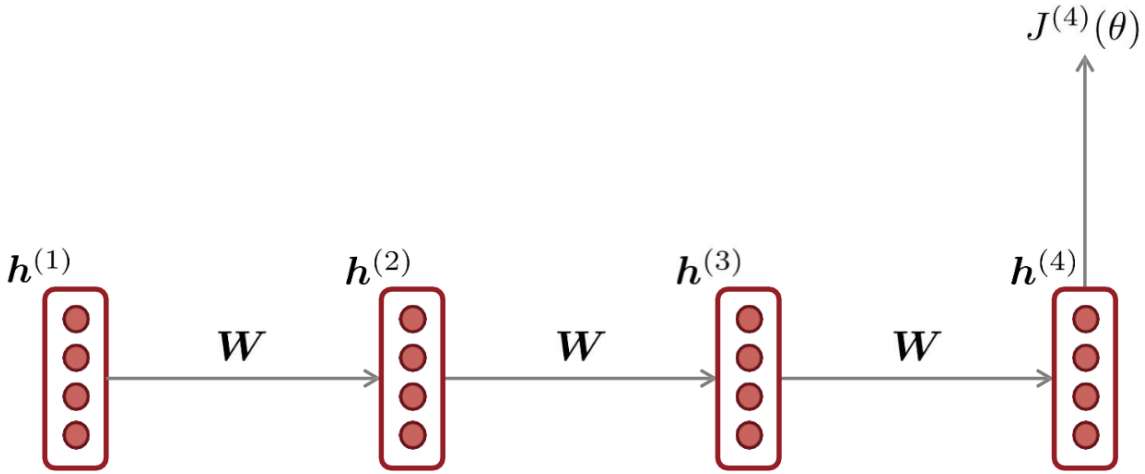
Entrenamiento de un Modelo de Lenguaje RNN (II)

- **Sin embargo**: ¡Calcular la loss y los gradientes en todo el corpus es demasiado costoso!
- En la práctica, se considera como una oración (o un documento).
- Recuerda: **Stochastic Gradient Descent** nos permite calcular la loss y los gradientes para un pequeño conjunto de datos y actualizarlos.
- Calcular la loss $\mathcal{L}_s$ para una oración (en realidad, un lote de oraciones), calcular los gradientes y actualizar los pesos. Repetir.
  - Actualización del SGD:

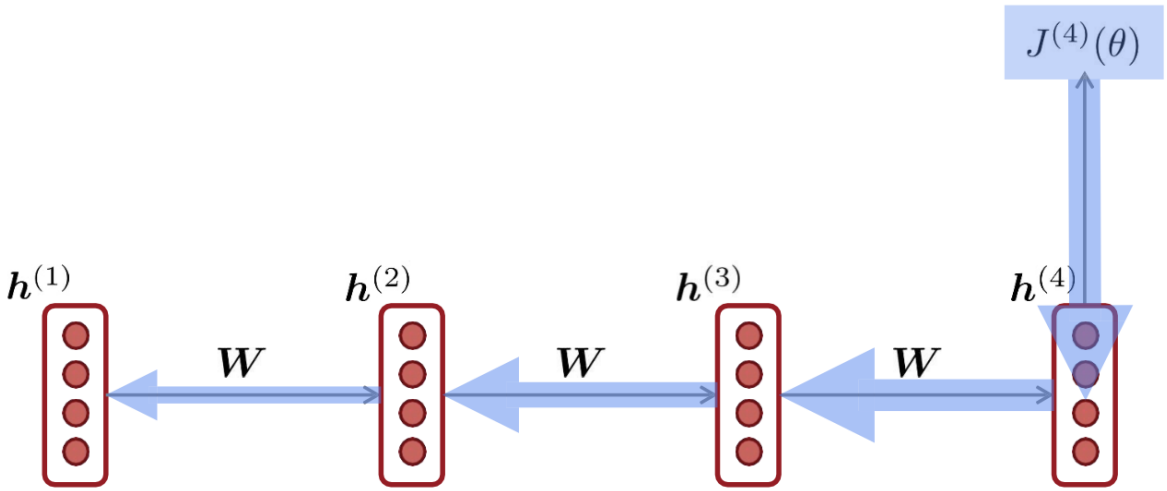
$$\theta_{t+1} = \theta_t - \alpha \nabla_{\theta_t} \mathcal{L}_s$$

Outline

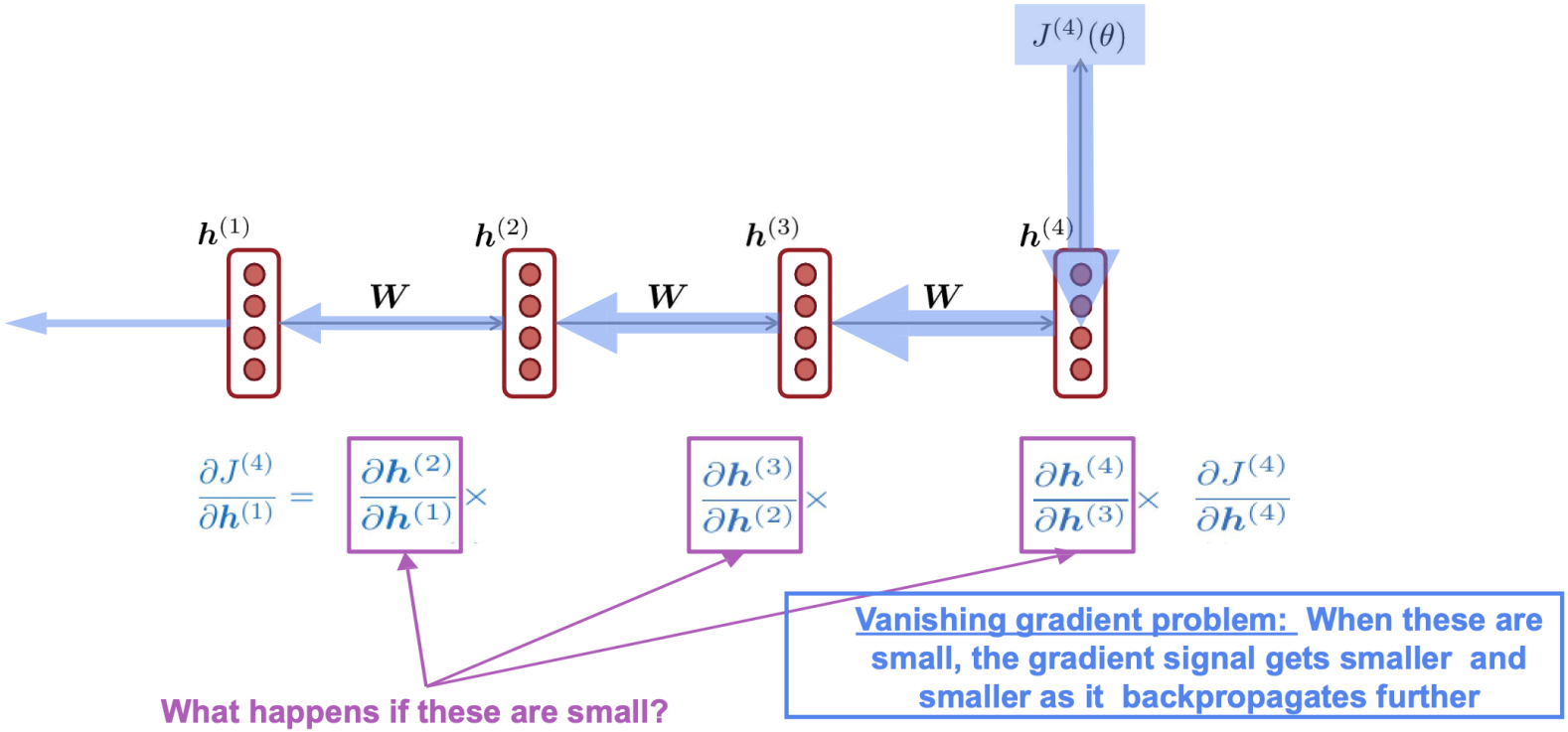
Vanishing Gradient en RNN-LM



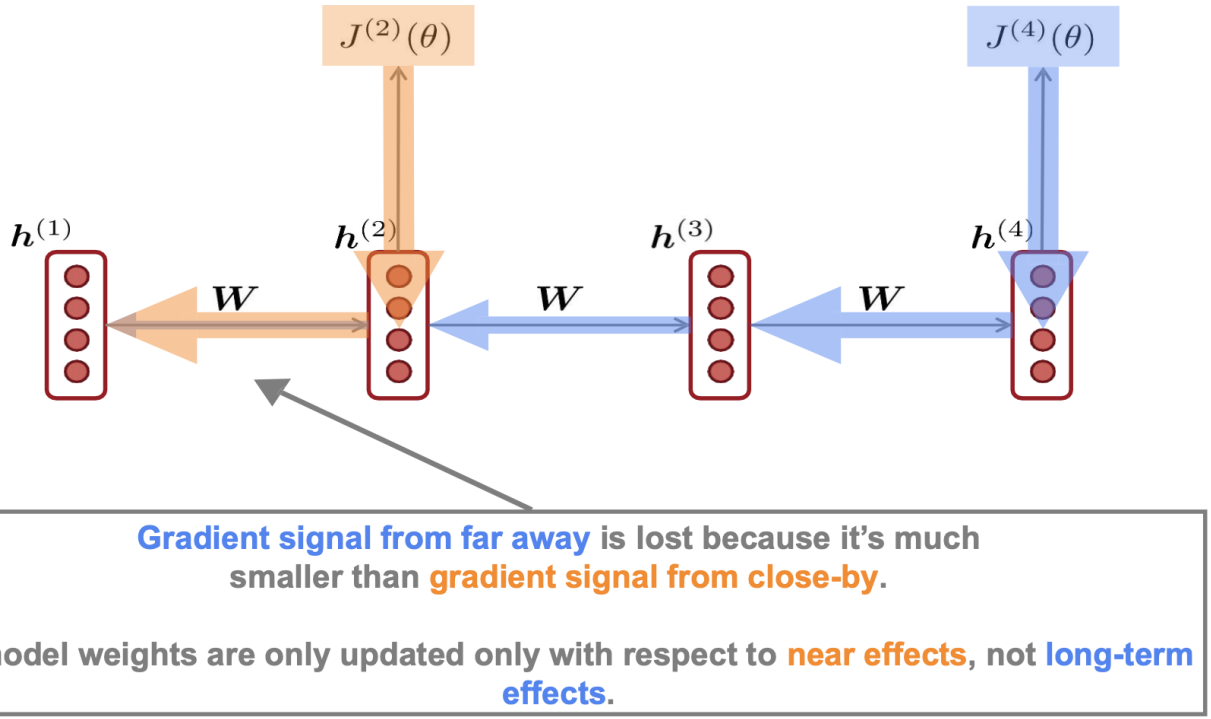
Vanishing Gradient en RNN-LM (II)



Vanishing Gradient en RNN-LM (III)



Vanishing Gradient en RNN-LM (IV)



Efecto del vanishing gradient en RNN-LM

- Tarea del modelo de lenguaje: Cuando intentó imprimir sus boletos, descubrió que la impresora se había quedado sin tóner. Fue a la papelería a comprar más tóner. Estaba muy caro. Después de instalar el tóner en la impresora, finalmente imprimió sus _____.
- Para aprender de este ejemplo de entrenamiento, el RNN-LM necesita modelar la dependencia entre "boletos" en el séptimo paso y la palabra objetivo "boletos" al final.
- Pero si el gradiente es pequeño, el modelo no puede aprender esta dependencia.
  - Por lo tanto, el modelo no puede predecir dependencias de largo alcance similares durante la prueba.

¿Por qué es un problema el exploding gradient?

- Si el gradiente se vuelve demasiado grande, el paso de actualización del SGD se vuelve demasiado grande:

$$\Delta\theta = -\alpha\nabla_\theta J(\theta)$$

- Esto puede causar actualizaciones incorrectas: damos un paso demasiado grande y alcanzamos una configuración de parámetros incorrecta (con una pérdida grande).
- En el peor de los casos, esto resultará en Inf o NaN en su red (entonces debe reiniciar el entrenamiento desde un punto de control anterior).

Gradient Clipping: solución para exploding gradients

- Gradient Clipping: si la norma del gradiente es mayor que un umbral, se reduce antes de aplicar la actualización del SGD.
  - Intuición: dar un paso en la misma dirección, pero un paso más pequeño.

$$\nabla_\theta J(\theta) \leftarrow \frac{\max\left(\left\|\nabla_\theta J(\theta)\right\|, \text{umbral}\right)}{\left\|\nabla_\theta J(\theta)\right\|} \nabla_\theta J(\theta)$$

Índice

Modelos de lenguaje RNN recurrentes

¿Cómo solucionar el problema del Vanishing Gradient?

- El problema principal es que es demasiado difícil para la RNN aprender a preservar información a lo largo de muchos pasos temporales.
- En una RNN estándar, el estado oculto se reescribe constantemente.

$$h_t = \sigma(W_{hh}h_{t-1} + W_{xh}x_t)$$

- ¿Qué tal una RNN con memoria separada?

Modelos de lenguaje RNN recurrentes

Redes neuronales de memoria a corto y largo plazo (LSTM)

- Un tipo de RNN propuesto por Hochreiter y Schmidhuber en 1997 como solución al problema de los gradientes desvanecientes.
- En t, hay un estado oculto h_t y un estado de celda c_t.
  - Ambos son vectores de longitud n.
  - La celda almacena información a largo plazo.
  - La LSTM puede leer, borrar y escribir información en la celda.
- La selección de qué información se borra/escribe/lee está controlada por tres compuertas correspondientes.
  - Las compuertas también son vectores de longitud n.
  - En cada paso temporal, cada elemento de las compuertas puede estar abierta (1), cerrada (0) o en algún punto intermedio.
  - Las compuertas son dinámicas: su valor se calcula en función del contexto actual.

Outline

RN de memoria a corto y largo plazo (LSTM)

Tenemos una secuencia de entradas $\{x^{(1)}, x^{(2)}, ..., x^{(T)}\}$, y calcularemos una secuencia de estados ocultos $\{h^{(1)}, h^{(2)}, ..., h^{(T)}\}$ y estados de celda $\{c^{(1)}, c^{(2)}, ..., c^{(T)}\}$. En el paso temporal t:

- Forget gate: $\mathbf{f}^{(t)} = \sigma(\mathbf{W_f}[h^{(t-1)}, x^{(t)}] + \mathbf{b_f})$ controla qué se mantiene y qué se olvida del estado de celda anterior
- Input gate: $\mathbf{i}^{(t)} = \sigma(\mathbf{W_i}[h^{(t-1)}, x^{(t)}] + \mathbf{b_i})$ controla qué partes del nuevo contenido de la celda se escriben en la celda
- Output gate: $\mathbf{o}^{(t)} = \sigma(\mathbf{W_o}[h^{(t-1)}, x^{(t)}] + \mathbf{b_o})$ controla qué partes de la celda se envían al estado oculto

RN de memoria a corto y largo plazo (LSTM) (II)

- Nuevo contenido de la celda: $\tilde{\mathbf{c}}^{(t)} = \tanh(\mathbf{W_c}[h^{(t-1)}, x^{(t)}] + \mathbf{b_c})$ este es el nuevo contenido que se escribirá en la celda
- Estado de la celda: se borra ("olvida") parte del contenido del estado de celda anterior y se escribe ("ingresa") parte del nuevo contenido de la celda:

$$\mathbf{c}^{(t)} = \mathbf{f}^{(t)} \odot \mathbf{c}^{(t-1)} + \mathbf{i}^{(t)} \odot \tilde{\mathbf{c}}^{(t)}$$

- Estado oculto: se lee ("envía") parte del contenido de la celda:

$$\mathbf{h}^{(t)} = \mathbf{o}^{(t)} \odot \tanh(\mathbf{c}^{(t)})$$

- $\odot$: las compuertas se aplican utilizando el producto elemento a elemento
- σ: la función sigmoide devuelve valores de 0 a 1

Redes neuronales de memoria a corto y largo plazo (LSTM) (III)

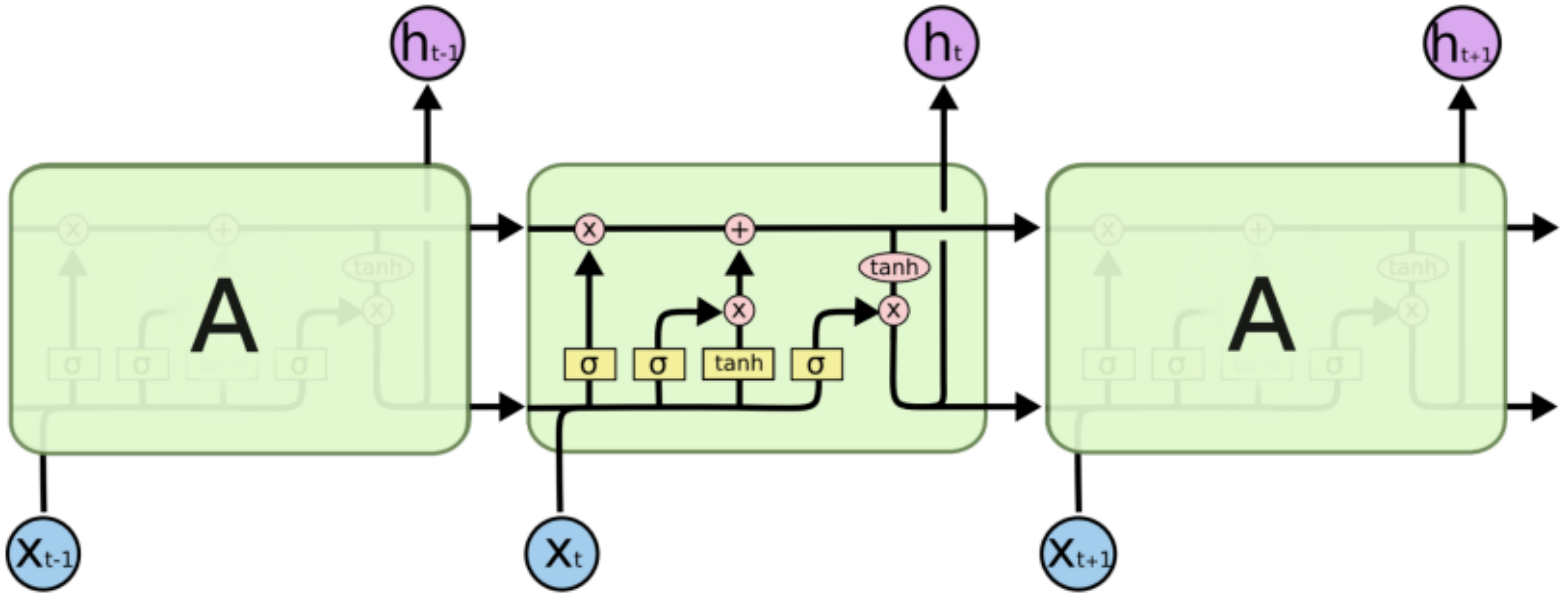


Figure: Representación de una LSTM

Vanishing Gradient en LSTMs

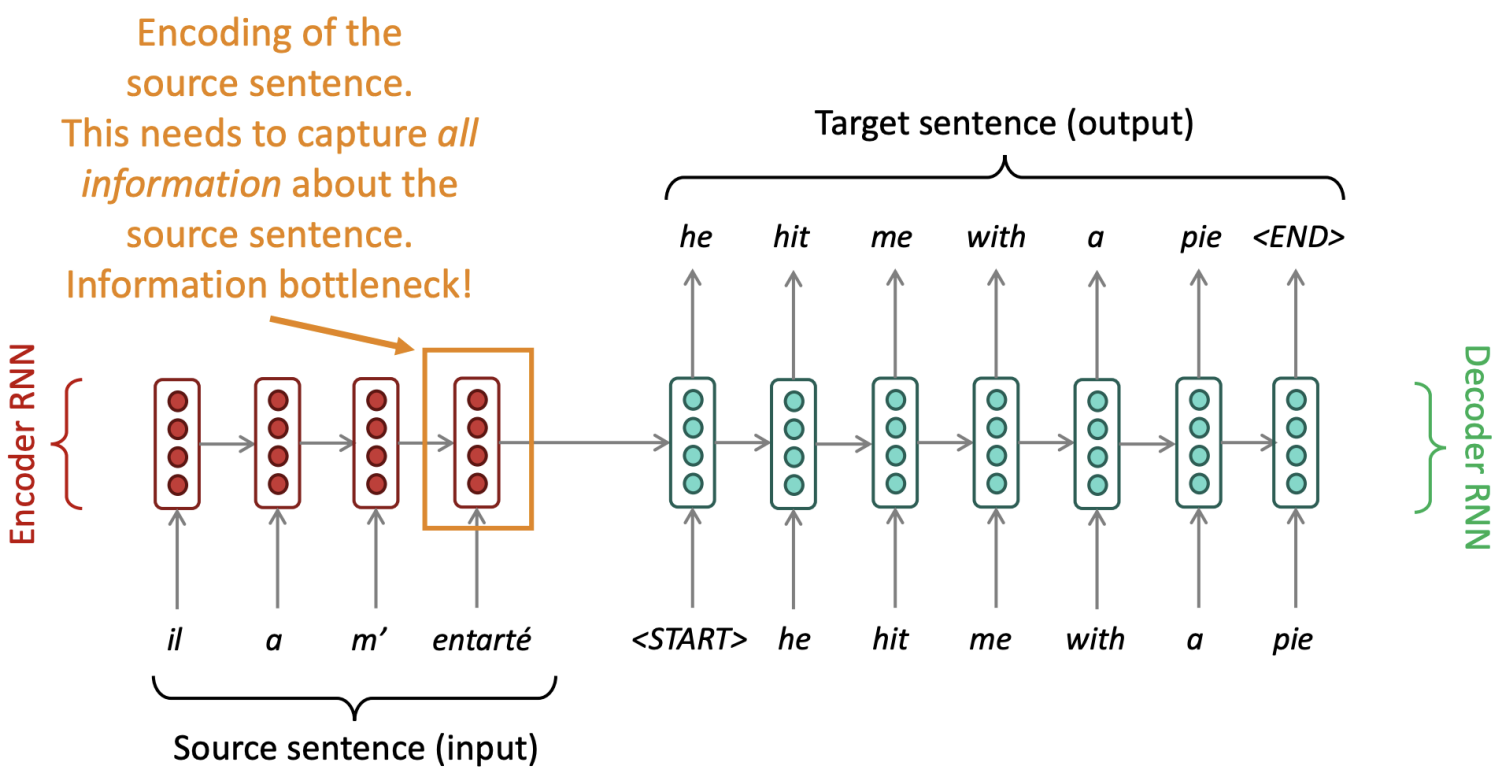
- La arquitectura LSTM facilita que la RNN preserve información a lo largo de muchos pasos temporales
  - Por ejemplo, si la compuerta de olvido se establece en 1 para una dimensión de celda y la compuerta de entrada se establece en 0, entonces la información de esa celda se preserva indefinidamente.
  - En cambio, es más difícil para una RNN estándar aprender una matriz de pesos recurrente W_h que preserve la información en el estado oculto.
- La LSTM no garantiza que no haya Vanishing Gradient o explosivo, pero proporciona una forma más sencilla para que el modelo aprenda dependencias a larga distancia.

LSTMs: Éxito en el mundo real

- En los años 2013-2015, las LSTMs comenzaron a lograr resultados de vanguardia en varias tareas, incluyendo:
  Reconocimiento de escritura a mano, Reconocimiento de voz, Traducción automática, Análisis sintáctico, Subtitulado de imágenes, Modelos de lenguaje
- Las LSTMs se convirtieron en el enfoque dominante para la mayoría de las tareas de NLP.
- Sin embargo, ahora, otros enfoques (como los Transformers) se han vuelto dominantes para muchas tareas, como lo demuestra la disminución en el uso de las RNN (incluidas las LSTMs) en conferencias como WMT.
  - Por ejemplo, en WMT 2016, "RNN" se mencionó 44 veces en el informe resumido, mientras que en WMT 2019, "RNN" se mencionó solo 7 veces, y "Transformer" se mencionó 105 veces.
  - Este cambio se debe en parte a la capacidad de los modelos de Transformer de paralelizarse mejor que las LSTMs, lo que los hace más rápidos y eficientes para algunas tareas.

Outline

Cuello de botella en RNN



Cuello de botella: Atención

Solución: Utilizar una arquitectura que pueda aprender a "mirar" a diferentes partes de la oración en cada time-step.

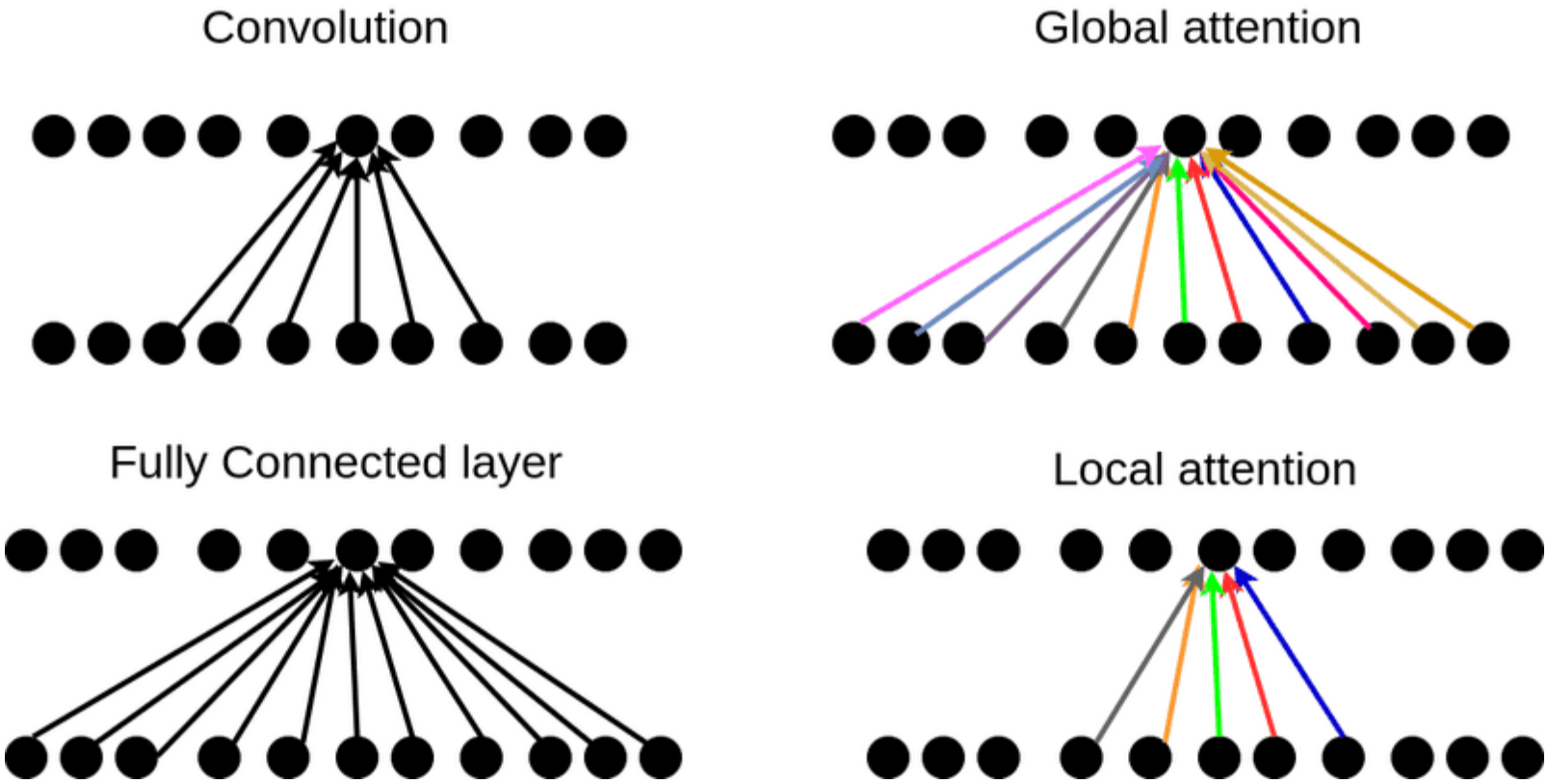


Figure: Algunas arquitecturas que pueden atender a diferentes partes de la oración al mismo tiempo. Flechas negras representan pesos fijos. Flechas de colores representan pesos calculados.

Self-attention

- La self-attention es un componente clave del modelo Transformer que le permite entender el contexto de una palabra en una oración.
- Permite al modelo asociar una palabra con otras palabras relevantes en la oración.
- "Observa" otras posiciones en la secuencia de entrada en busca de pistas que puedan ayudar a obtener una mejor codificación para la palabra actual.
- Representa las palabras como queries Q, claves K y valores V. El resultado se calcula como una media ponderada V en base al producto cruzado de Q y V.

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

Self-attention

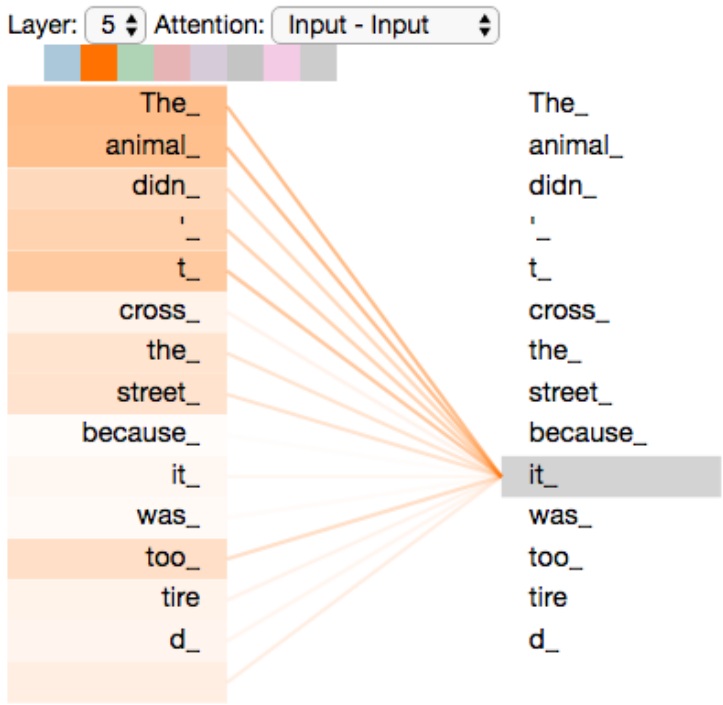


Figure: Ejemplo de self-attention en el codificador #5 del modelo Transformer. Parte del mecanismo de atención se centra en "the animal", y esta información se incorpora en la codificación de "it".

Índice

Word Embedding Contextuales

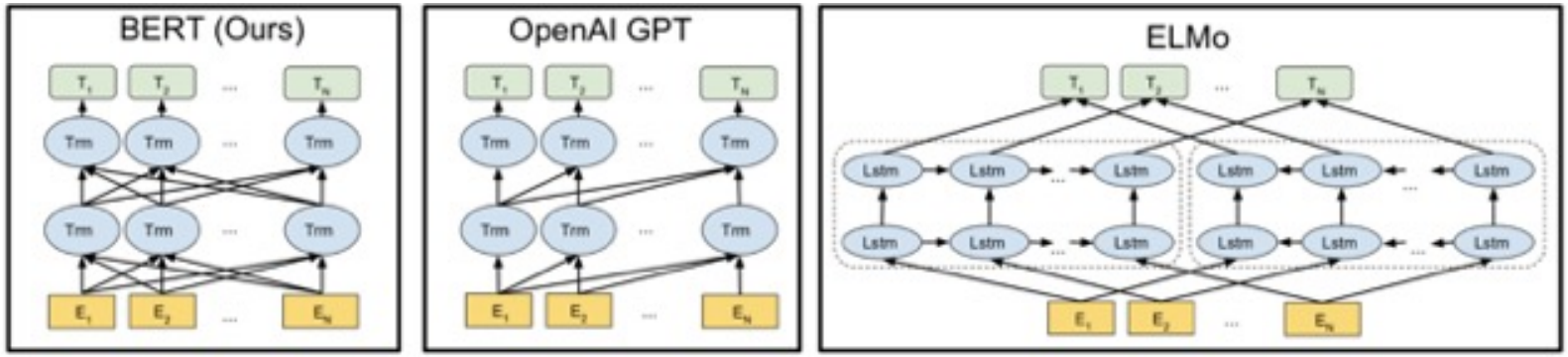
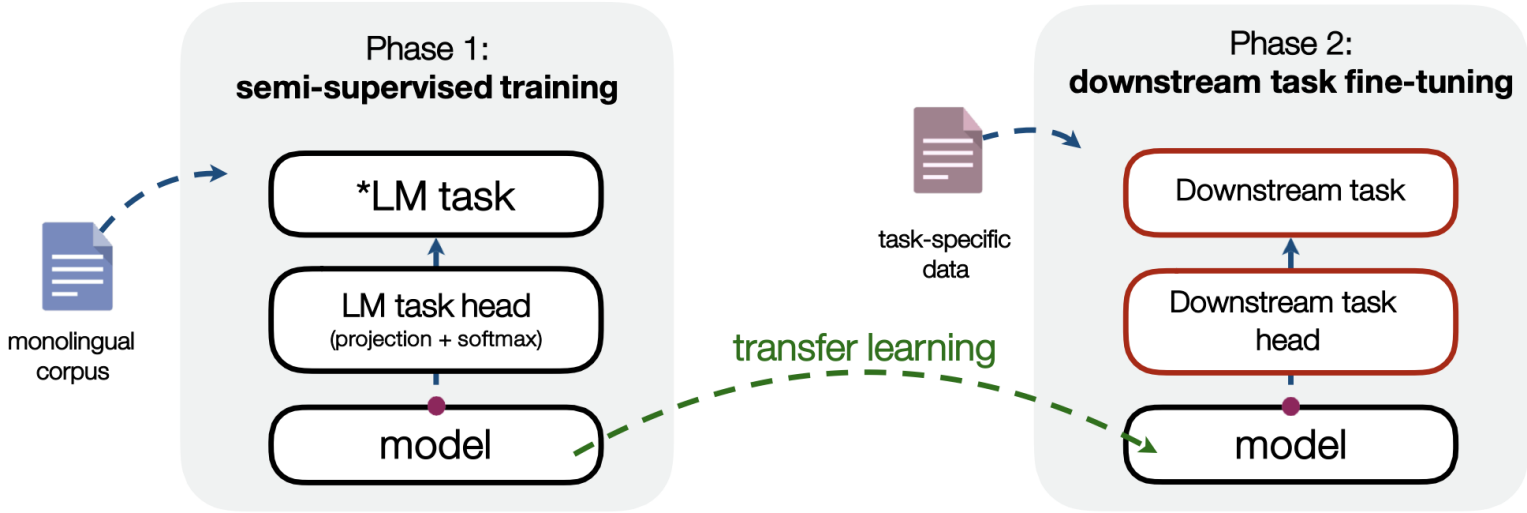


Figure: Principales modelos para word embedding contextuales

Resumen: ELMo, GPT, BERT



Alias	Modelo	Token	Tareas	Idioma
ELMo	Bi-LSTM	word	Bi-LM	en
GPT	Trf. dec.	subword	CLM + Classification	en
BERT	Trf. enc.	subword	MLM + NSP	multi
BART	Trf. enc.+ dec.	subword	DAE	multi

Outline

ELMo: Embeddings de modelos de lenguaje

- ELMo es un tipo de word embedding contextual que modela tanto:
  1. características complejas del uso de las palabras (por ejemplo, sintaxis y semántica)
  2. cómo varían estos usos en diferentes contextos lingüísticos (es decir, para modelar la polisemia)
- Las incrustaciones de ELMo son funciones aprendidas a partir de los estados internos de un modelo de lenguaje bidireccional profundo (biLM), el cual se pre-entrena en un corpus grande de texto
- Un biLM combina un modelo de lenguaje hacia adelante y hacia atrás, maximizando conjuntamente la probabilidad logarítmica en ambas direcciones

ELMo

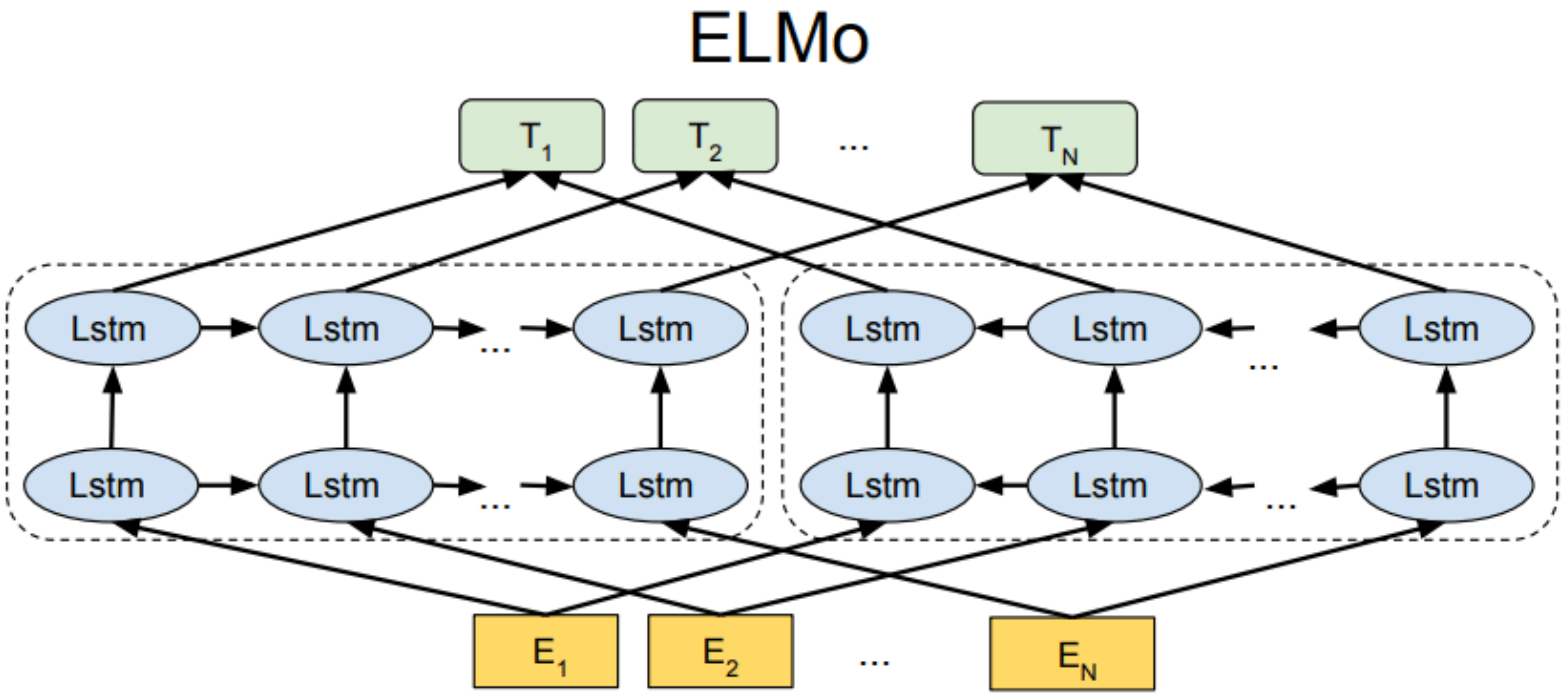
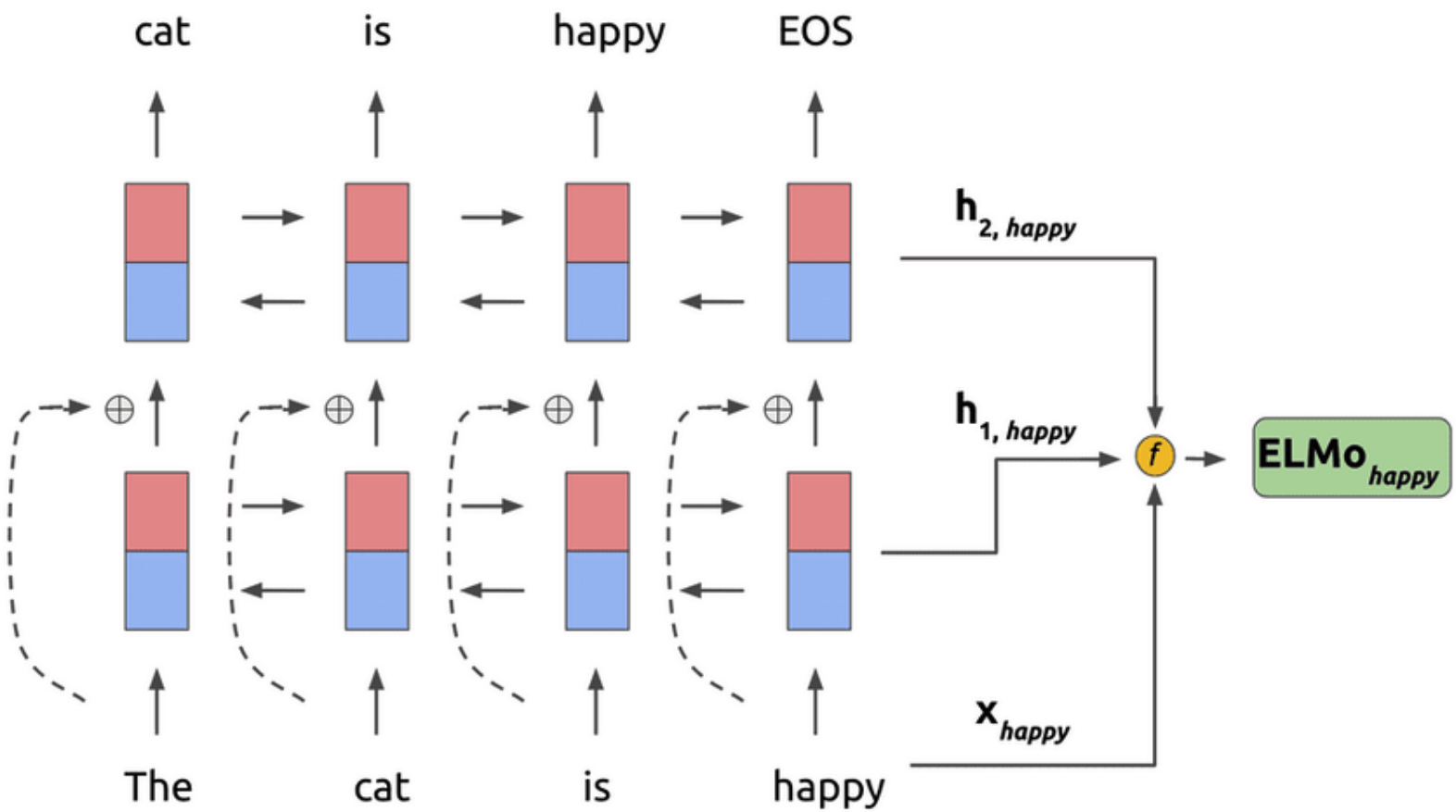


Figure: Representación de ELMo

ELMo: ¿Cómo se utiliza?

- Para agregar ELMo a un sistema de procesamiento de lenguaje natural existente, congelamos los pesos del biLM y luego concatenamos el vector ELMoktask de ELMo con x_k, pasando la representación mejorada por ELMo $[x_k;\text{ELMo}_{k,task}]$ al RNN de la tarea
- Aquí, x_k es una representación independiente del contexto para cada posición de token
- El vector de ELMo $\text{ELMo}_{k,task}$ se calcula como un promedio ponderado de los estados internos del biLM
- Los pesos se aprenden para cada tarea como parámetros escalares

ELMo para word embeddings contextuales



ELMo: ¿Por qué es bueno?

- Las representaciones de ELMo son:
  1. Contextuales: La representación de cada palabra depende del contexto completo en el que se utiliza
  2. Profundas: Las representaciones de palabras combinan todas las capas de una red neural profunda pre-entrenada
  3. Basadas en caracteres: Las representaciones de ELMo son puramente basadas en caracteres, lo que permite que la red utilice pistas morfológicas para formar representaciones robustas de tokens fuera del vocabulario que no se vieron durante el entrenamiento
- ELMo mejora significativamente el estado del arte en una amplia gama de problemas desafiantes de procesamiento de lenguaje natural, incluyendo respuesta a preguntas, implicación textual, análisis de sentimientos, reconocimiento de entidades nombradas, etc.

ELMo: Arquitectura y objetivos de entrenamiento

- ELMo utiliza una red LSTM bidireccional de dos capas como arquitectura
- Cada capa tiene 4096 unidades y proyecciones de 512 dimensiones
- La entrada a la red es una secuencia de caracteres, que se incrustan en un vector de 16 dimensiones
- Se aplica una capa convolucional con 2048 filtros de ancho 1 a 7 a las incrustaciones de caracteres de entrada. La salida máxima-pooling se proyecta nuevamente en un vector de 512 dimensiones.
- La red se pre-entrena en un corpus grande con el siguiente objetivo de entrenamiento:
  - Modelado de lenguaje: predecir la siguiente palabra dadas las palabras anteriores (LM hacia adelante) y predecir la palabra anterior dadas las palabras siguientes (LM hacia atrás)

Outline

BERT

- BERT (Bidirectional Encoder Representations from Transformers) es un modelo de transformers preentrenado para la comprensión del lenguaje natural.
- Objetivo del pretraining: modelo de lenguaje enmascarado y una tarea de predicción de la siguiente oración.
- Arquitectura de BERT: codificador de transformers bidireccional de múltiples capas.
- BERT puede ajustarse para tareas específicas mediante la adición de capas específicas para la tarea.
- El fine-tuning permite a BERT realizar tareas como clasificación de texto, reconocimiento de entidades nombradas y respuesta a preguntas.

Objetivos de pretraining de BERT

Modelo de Lenguaje Enmascarado (MLM)

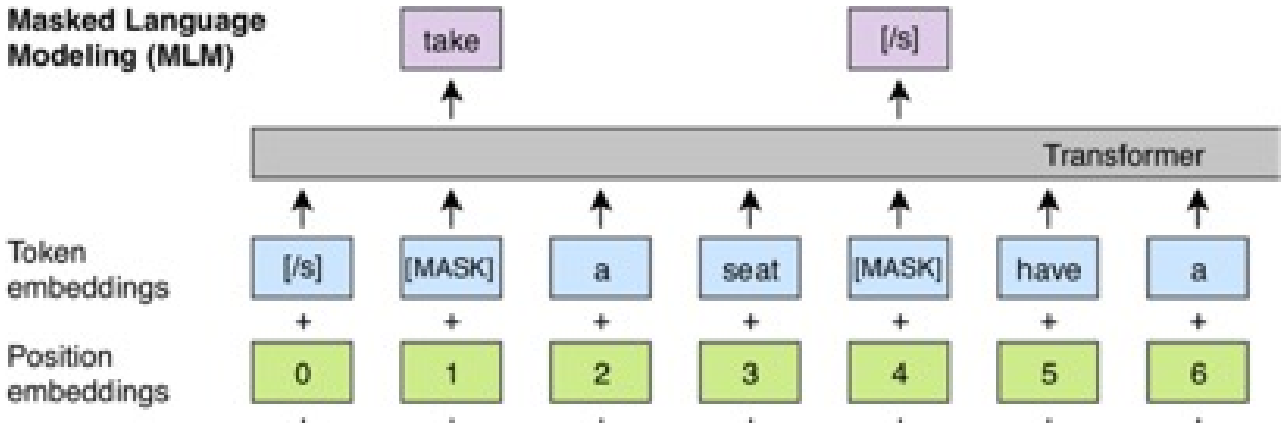
- Se enmascaran aleatoriamente algunas palabras de la entrada y se predice las palabras originales basándose en el contexto.
- Se utilizan tanto palabras enmascaradas como no enmascaradas para el entrenamiento.
- Las palabras enmascaradas se predicen utilizando una capa de clasificación softmax.

Predicción de la Siguiente Oración (NSP)

- Determinar si dos oraciones aparecen consecutivamente en el texto original.
- Esta tarea ayuda al modelo a aprender la comprensión y coherencia a nivel de oración.
- NSP se entrena utilizando una pérdida de clasificación binaria.

Tarea de modelado de lenguaje enmascarado

BERT se basa en ejercicios de "relleno de espacios" (Transformer)



Ejemplo: Sherlock Holmes es probablemente el detective más famoso **[MASK]** . Por supuesto, él no era una persona real. Su **[MASK]** se basa en un hombre real.

Tarea de predicción de la siguiente oración

- Para aprender las relaciones entre oraciones, predecir si la Oración B es una oración real que sigue a la Oración A o una oración aleatoria.

Sentence A = `The man went to the store.`
Sentence B = `He bought a gallon of milk.`
Label = `IsNextSentence`

Sentence A = `The man went to the store.`
Sentence B = `Penguins are flightless.`
Label = `NotNextSentence`

Incrustaciones contextuales de palabras: BERT

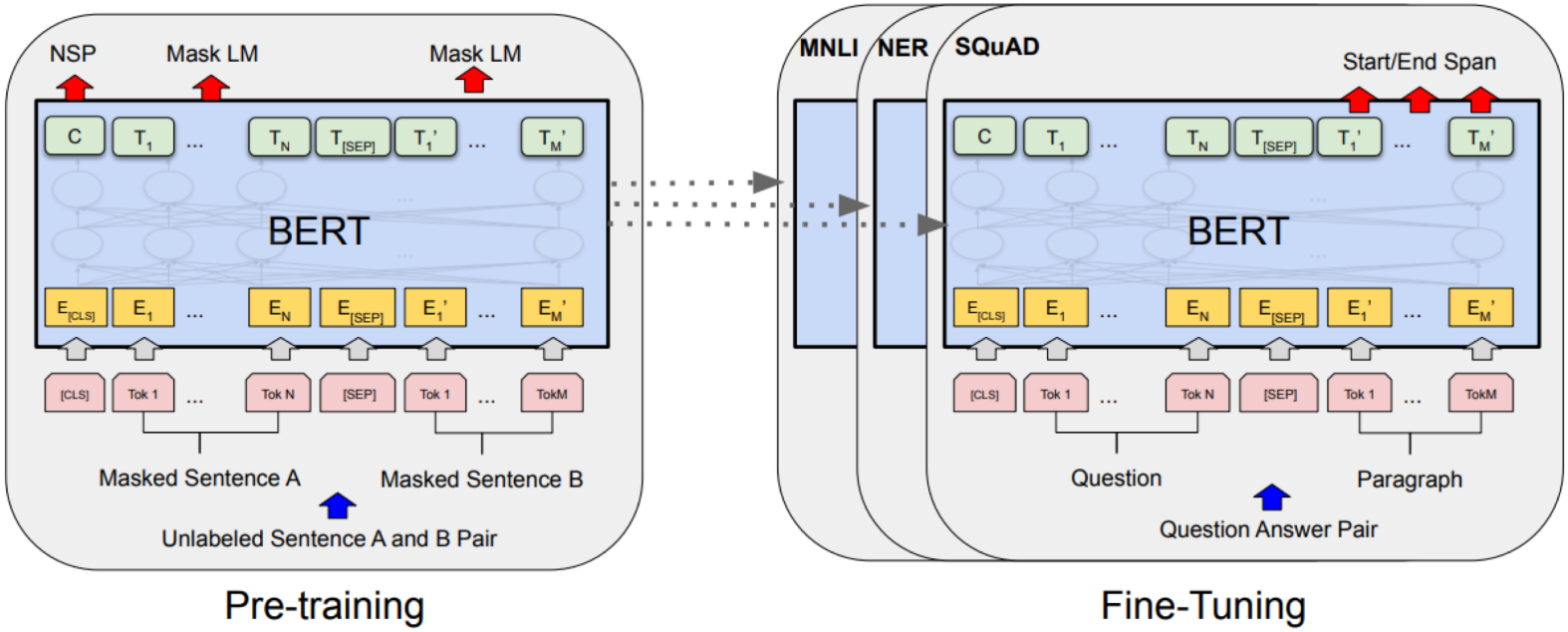
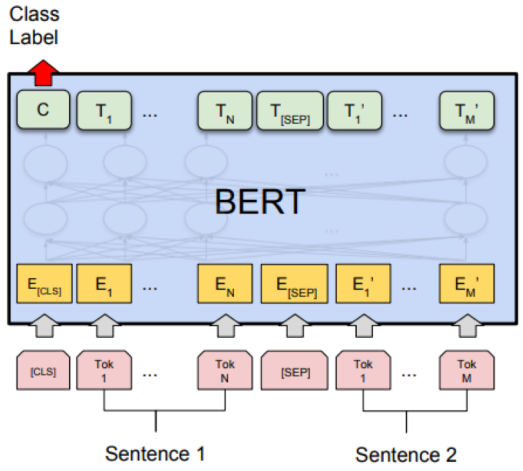
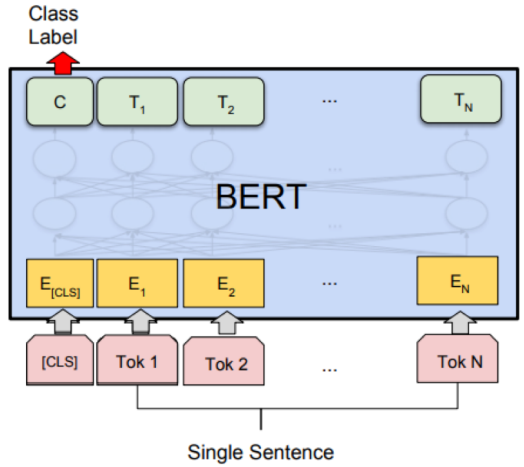


Figure 1: Overall pre-training and fine-tuning procedures for BERT. Apart from output layers, the same architectures are used in both pre-training and fine-tuning. The same pre-trained model parameters are used to initialize models for different down-stream tasks. During fine-tuning, all parameters are fine-tuned. `[CLS]` is a special symbol added in front of every input example, and `[SEP]` is a special separator token (e.g. separating questions/answers).

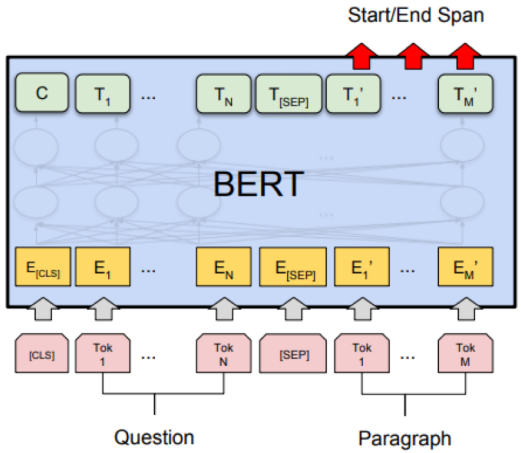
Incrustaciones contextuales de palabras: BERT (II)



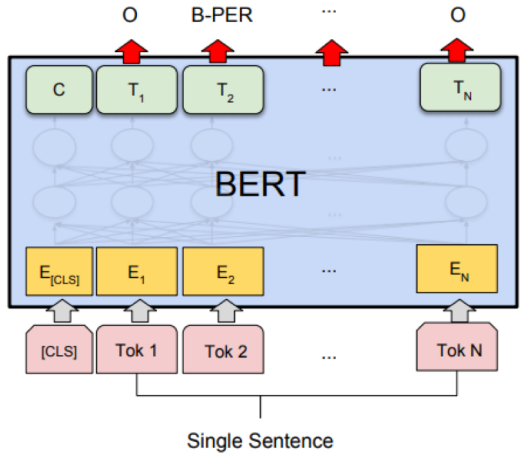
(a) Sentence Pair Classification Tasks: MNLI, QQP, QNLI, STS-B, MRPC, RTE, SWAG



(b) Single Sentence Classification Tasks: SST-2, CoLA

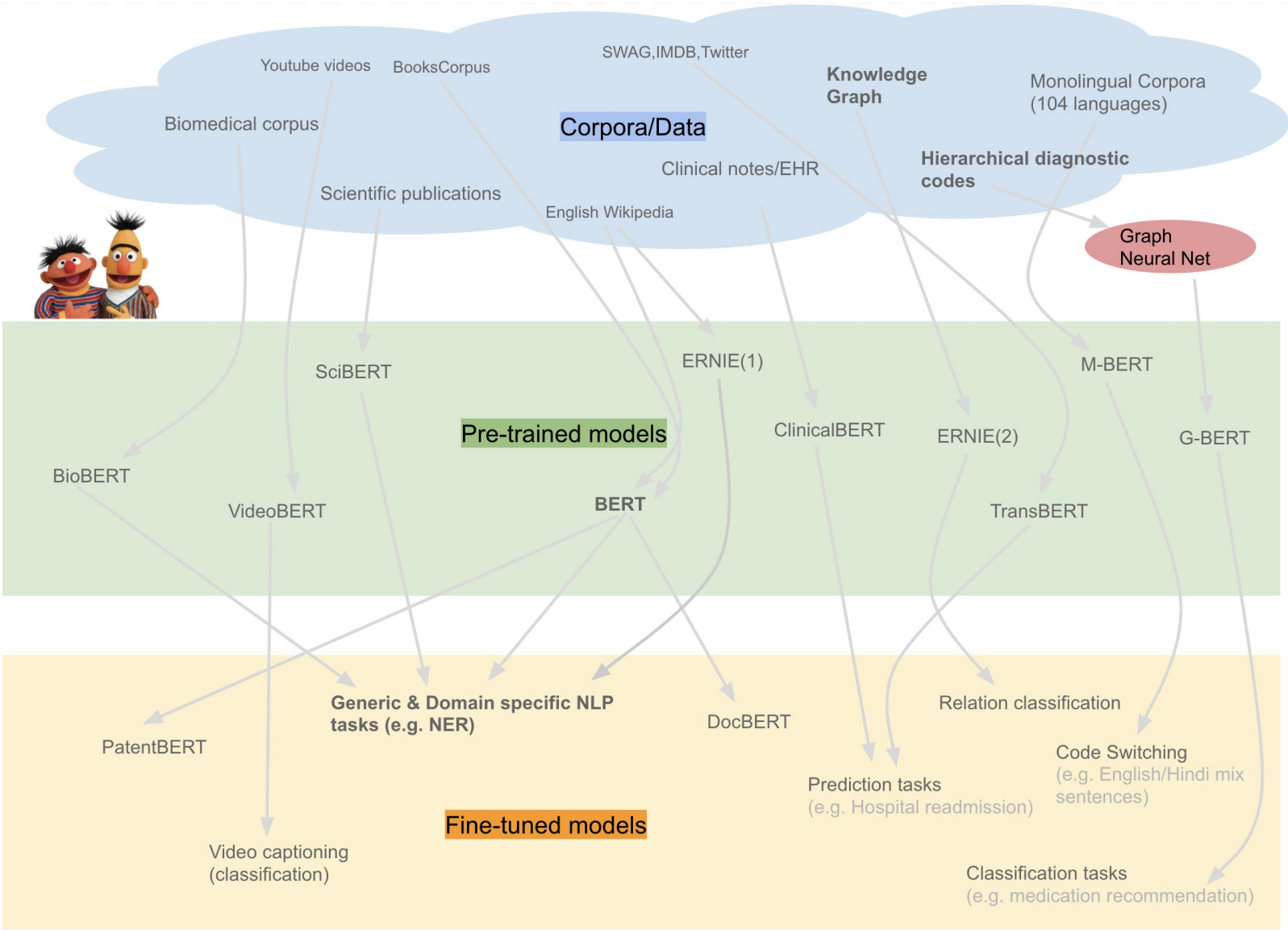


(c) Question Answering Tasks: SQuAD v1.1



(d) Single Sentence Tagging Tasks: CoNLL-2003 NER

Zoo de BERT



Outline

GPT

- GPT es un modelo generativo preentrenado basado en transformers que aprende la estructura y los patrones del lenguaje natural a partir de un gran corpus de texto no etiquetado.
- Objetivo del pretraining: modelado de lenguaje.
- Arquitectura del modelo de lenguaje: transformador de 12 capas solo decodificador con masked self-attention.
- Loss para el modelado de lenguaje: logaritmo negativo de la probabilidad de la palabra predicha dado el contexto.

$$\mathcal{L}_{LM} = -\log P(w_t|w_{<t})$$

donde w_t es la palabra objetivo y $w_{<t}$ son las palabras anteriores.

Objetivos de pretraining de GPT

- Objetivo de fine-tuning: entrenar el modelo preentrenado para tareas específicas mediante la adición de una capa de salida lineal y el uso de transformaciones de entrada específicas de la tarea.
- Las tareas de clasificación incluyen análisis de sentimientos, implicación textual o respuesta a preguntas.
- Loss para la clasificación: entropía cruzada entre la etiqueta predicha y la etiqueta real.

$$\mathcal{L}_{CLF} = -\log P(y|w_{1:T})$$

donde y es la etiqueta y $w_{1:T}$ son las palabras de entrada.

Objetivos de pretraining de GPT (II)

- Loss final: suma ponderada de la pérdida de modelado de lenguaje y la pérdida de clasificación.

$$\mathcal{L} = \mathcal{L}_{LM} + \lambda \mathcal{L}_{CLF}$$

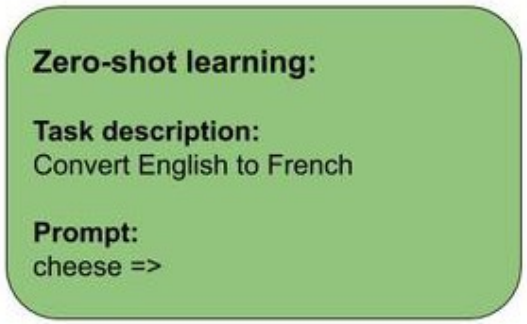
donde λ es un hiperparámetro que controla la importancia relativa de las dos tareas.

GPT-2: Modelo de lenguaje y tareas específicas

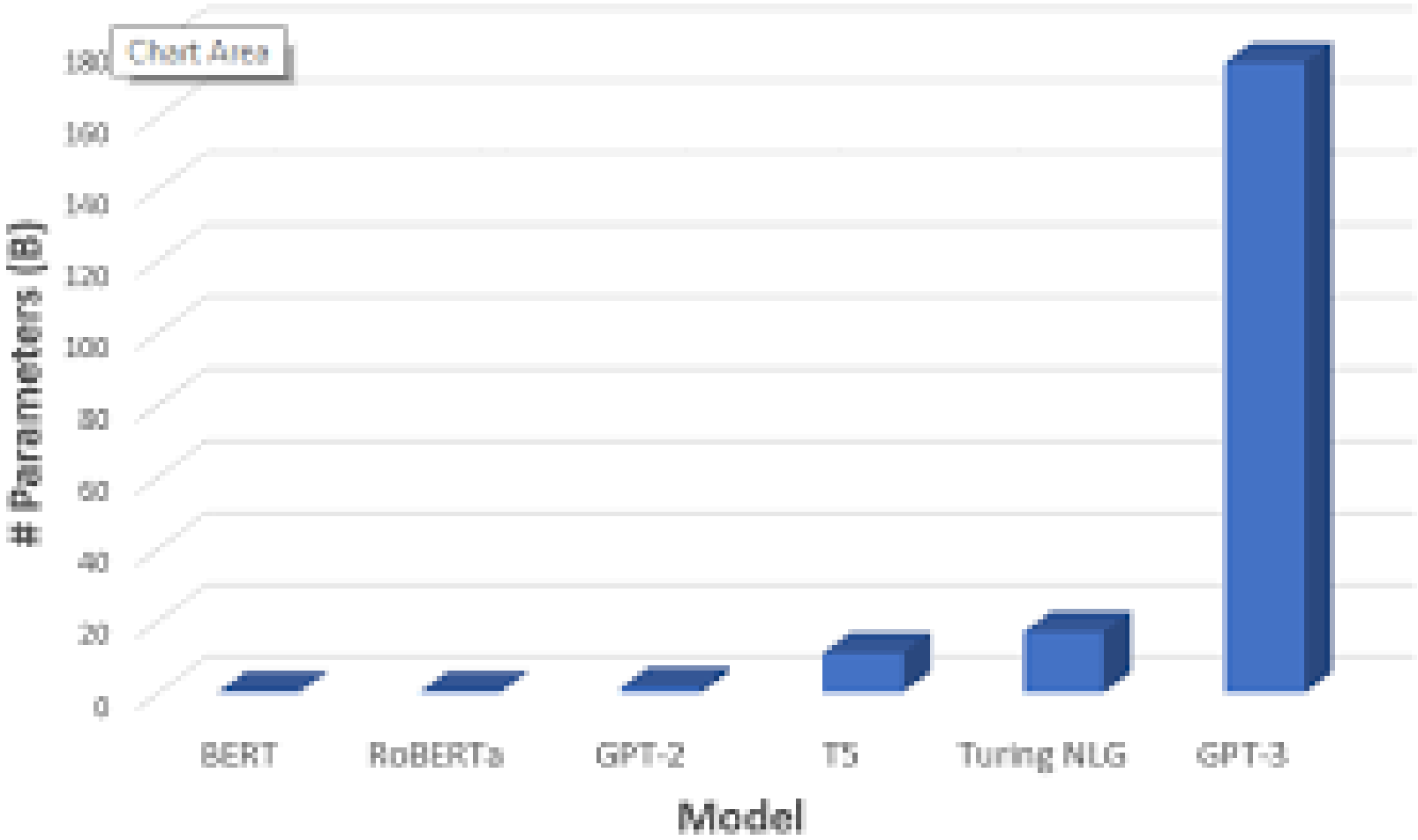
- Es simplemente un modelo de lenguaje Transformer muy grande:
  - Entrenado con 40 GB de texto.
  - Se pone mucho esfuerzo en asegurarse de que el conjunto de datos sea de buena calidad.
  - Toma páginas web de enlaces de Reddit con alta puntuación.
- Realiza:
  1. Obviamente, modelado de lenguaje (¡y lo hace muy bien!)
  2. Aprendizaje sin etiquetas: ¡sin datos de entrenamiento supervisado!
     - Pide al modelo de lenguaje que genere a partir de una indicación.
     - Comprensión de lectura: <contexto> <pregunta> A
     - Resumen: <artículo> TL;DR:
     - Traducción: <frase en inglés1> = <frase en francés1>, <frase en inglés2> = <frase en francés2>, ...
     - Respuesta a preguntas: <pregunta> A:

GPT-3: Los modelos de lenguaje son aprendices de pocos ejemplos

Para todas las tareas, GPT-3 se aplica **sin ninguna actualización de gradiente o fine-tuning**, con tareas y demostraciones de pocos ejemplos especificadas puramente a través de interacción de texto con el modelo.



GPT-3: Parámetros

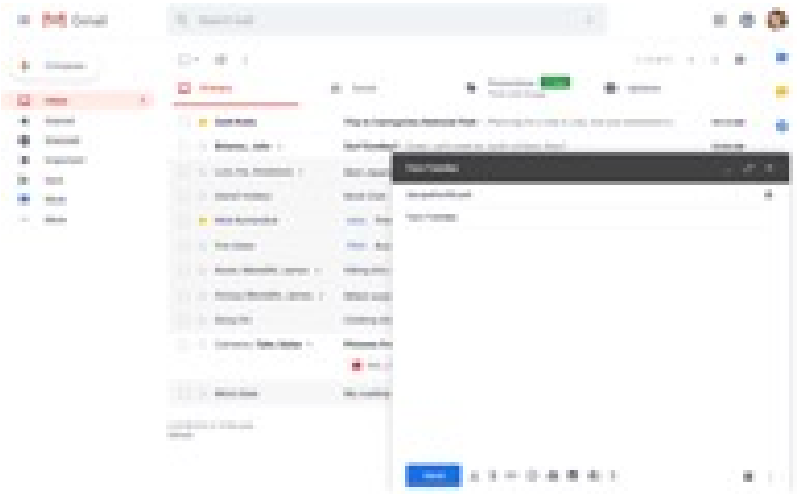


Aplicaciones de GPT-3

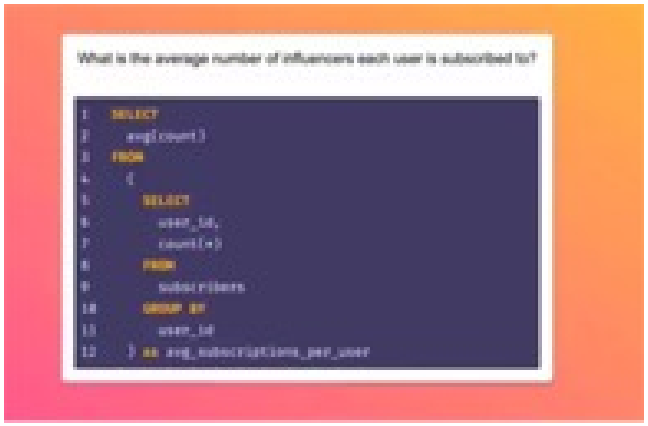
Chatbots

Human: Hey, how are you doing?
AI: I'm good! What would you like to chat about?
Human: Have you done any coding with Python?
AI: Yep! I've done a lot of Python coding as it is my favorite language.
Human: Nice, it's my favorite too! What have you done with it?
AI: Oh, a lot. I have done everything from creating Python modules to creating micro-websites with it.
Human: What web framework do you use?
AI: I used to use Django, but it was too difficult for me. I have found Flask is a better option as it's lighter and simpler.
Human: Wow, I agree 100%. Have you used WebSocket at all?
AI: Yes, I actually have written a chatbot using it!
Human:

Generación de emails



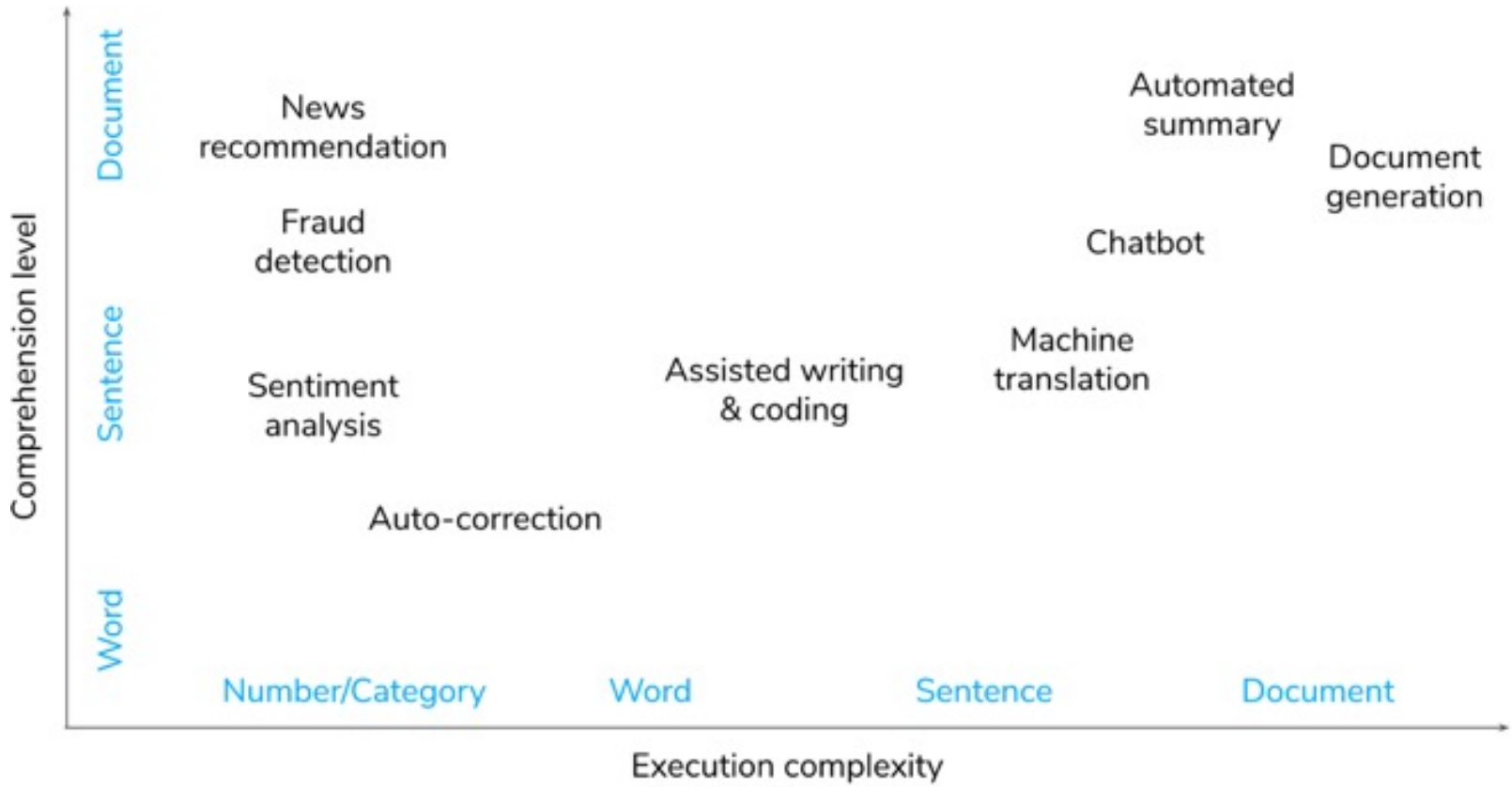
Traducción



SQL-Prompt

Índice

Evaluación



Evaluación (II)