Redes Neuronales para Lenguaje Natural

2023

Grupo de Procesamiento de Lenguaje Natural
Instituto de Computación


Transformer

Jurafsky and Martin 3rd edition. Caps 10,11. https://web.stanford.edu/~jurafsky/slp3/

Machine Learning with PyTorch and Scikit-Learn. Raschka et al. 2022. Cap 16.

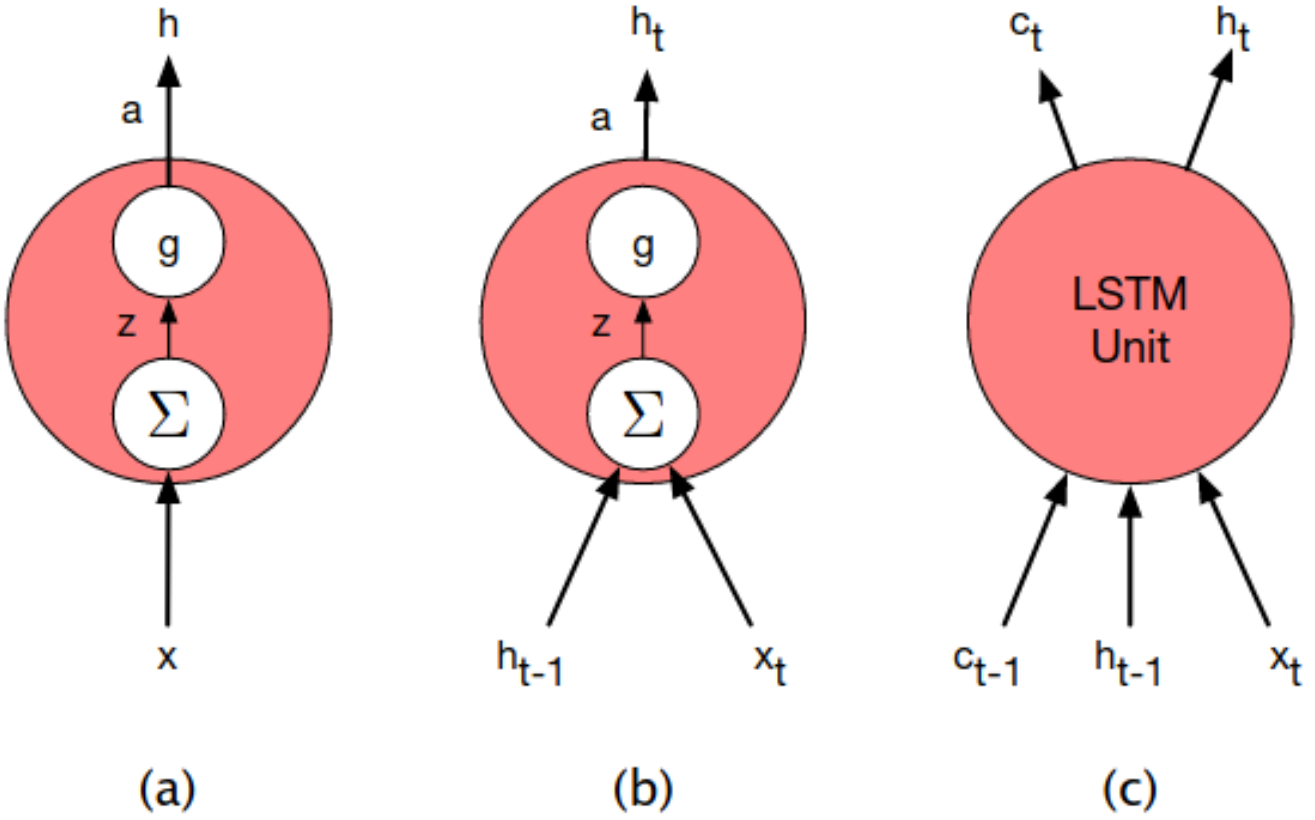
Transformer

- Multihead Self-attention
- Positional Embeddings
- Transformer Block
- LM with Transformer
- Transformer for generation
- Pretraining and Fine-Tuning

Motivación y repaso

Para procesar **secuencias** (como el lenguaje) y vimos:

- **Multilayer Perceptron** (o red completamente conectada, o densa)
  - **Centroide** (simple, funciona demasiado bien, pierde el orden)
  - **Ventana** (contexto muy limitado, funciona en algunos casos)
- **Convolutional Neural Networks** (CNNs)
- **Recurrent Neural Networks** (RNNs)
  - Usar los estados ocultos tienen una representación “condensada”
    - desvanecimiento o explosión de gradiente
    - LSTMs, GRUs → Mejoran el manejo del contexto
- **RNN + Attention**



Extraído de https://web.stanford.edu/~jurafsky/slp3/

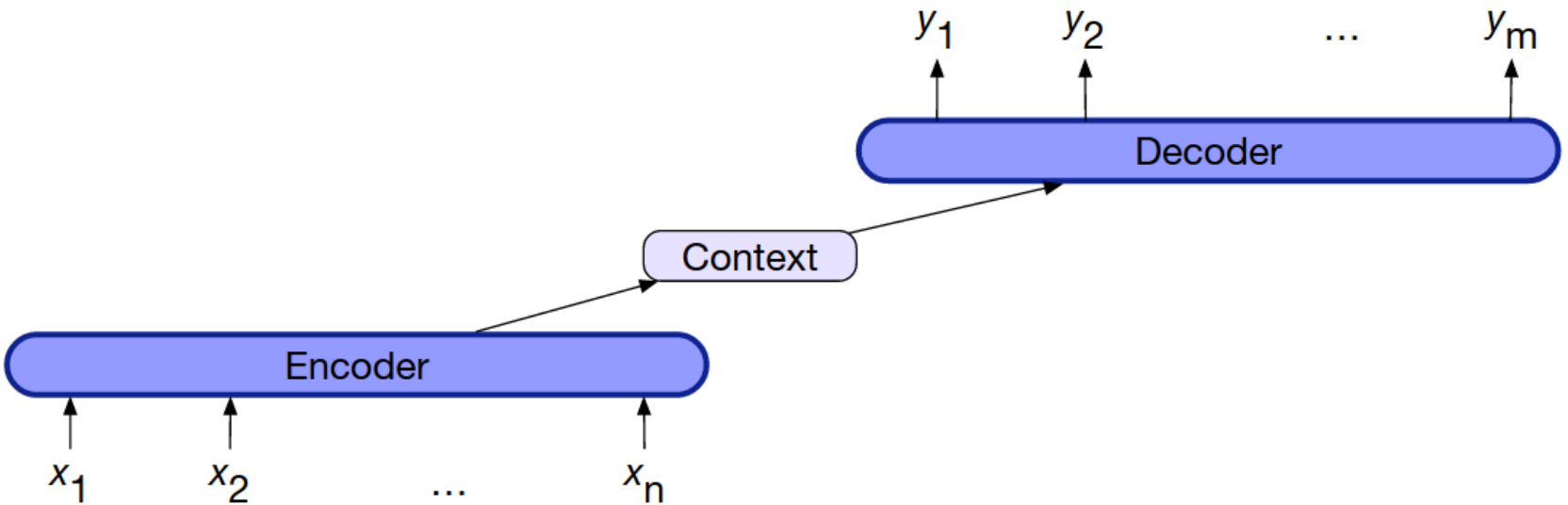
ELMo (representaciones contextualizadas)

Peters, Matthew E. et al. “Deep Contextualized Word Representations.” NAACL (2018).

ULMFiT (fine-tuning)

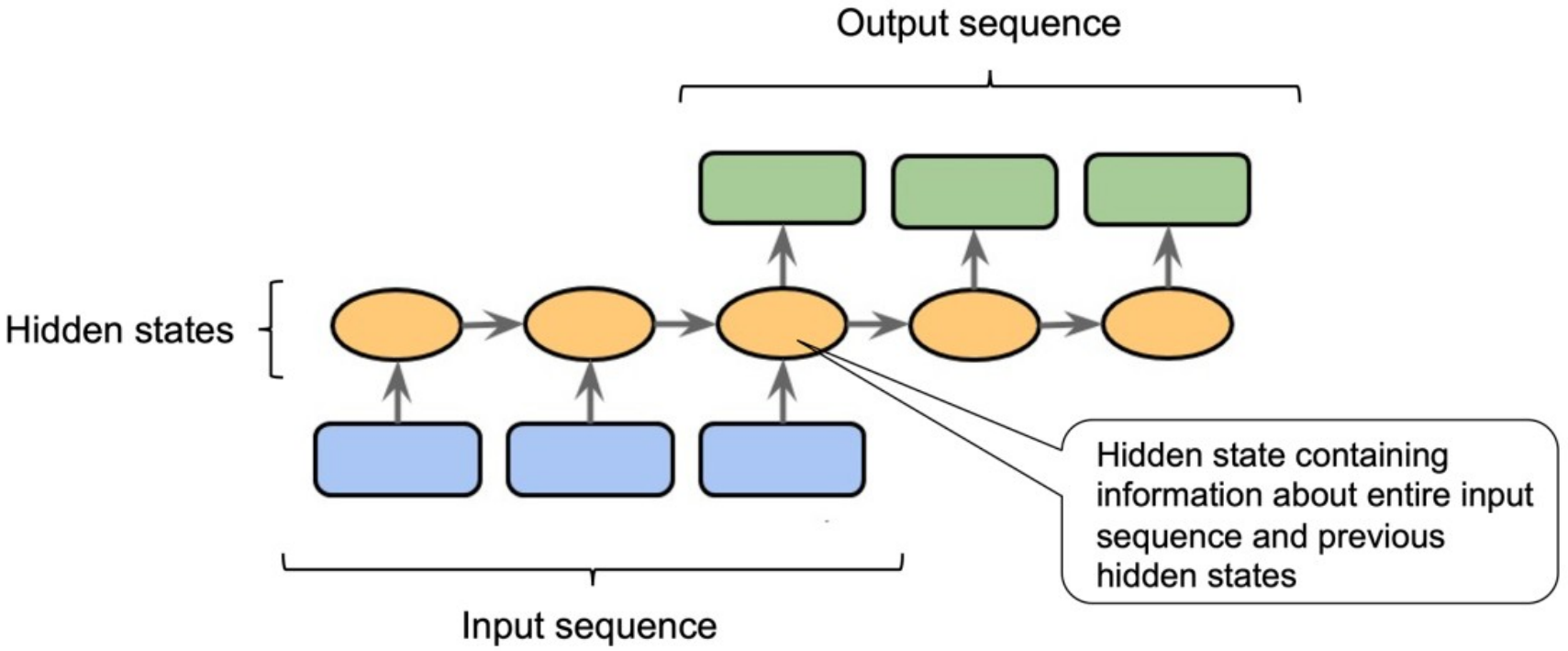
Howard, Jeremy and Sebastian Ruder. “Fine-tuned Language Models for Text Classification.” ArXiv abs/1801.06146 (2018)

Encoder-decoder

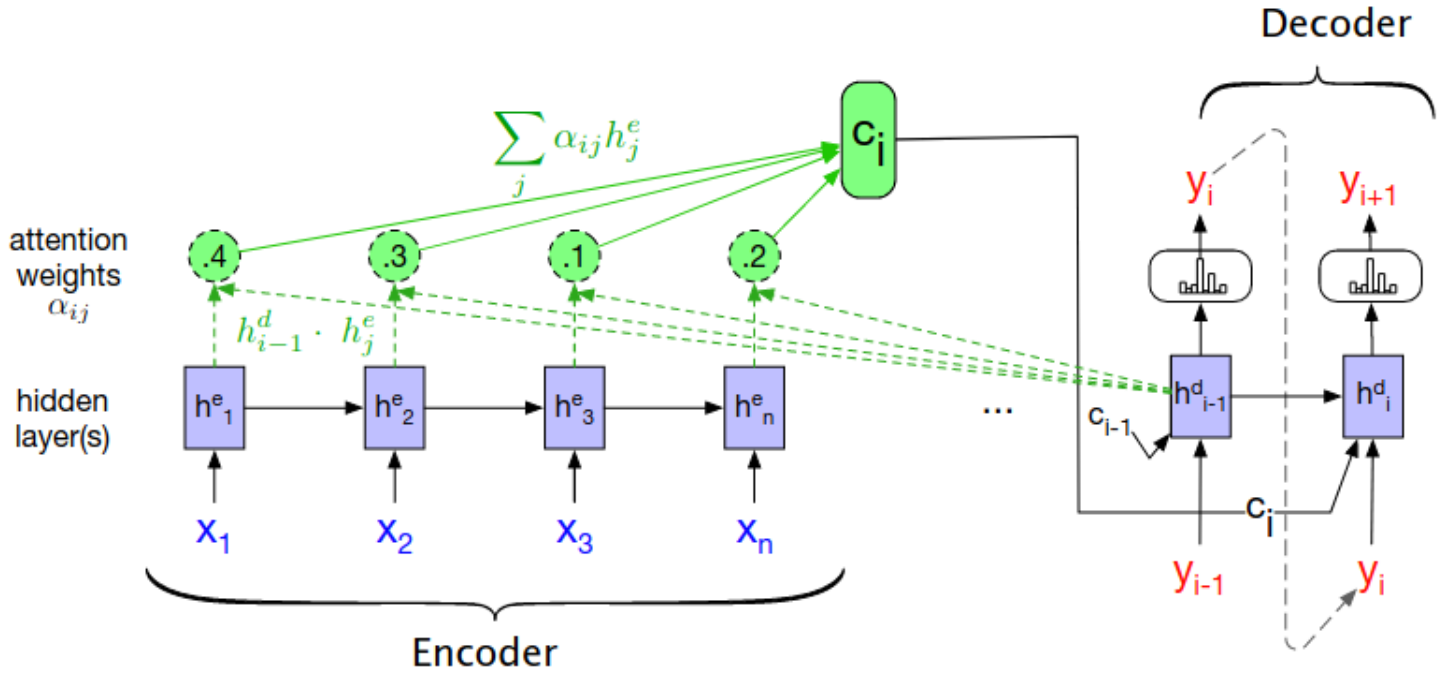


Basado en https://web.stanford.edu/~jurafsky/slp3/

Encoder-decoder (RNN)



Encoder-decoder (RNN + Attention)



Encoder-Decoder (Seq2Seq)

- Traductores automáticos,
- Resumen abstractivo,
- Generative Question-answering,
- Generative chatbots,
    - prompting
        - ¿resumen extractivo? ¿QA extractivo?

Transformer

Attention Is All You Need

Ashish Vaswani* Google Brain avaswani@google.com

Noam Shazeer* Google Brain noam@google.com

Niki Parmar* Google Research nikip@google.com

Jakob Uszkoreit* Google Research usz@google.com

Llion Jones* Google Research llion@google.com

Aidan N. Gomez* † University of Toronto aidan@cs.toronto.edu

Łukasz Kaiser* Google Brain lukaszkaiser@google.com

Illia Polosukhin* ‡ illia.polosukhin@gmail.com

Abstract

The dominant sequence transduction models are based on complex recurrent or convolutional neural networks that include an encoder and a decoder. The best performing models also connect the encoder and decoder through an attention mechanism. We propose a new simple network architecture, the Transformer, based solely on attention mechanisms, dispensing with recurrence and convolutions entirely. Experiments on two machine translation tasks show these models to be superior in quality while being more parallelizable and requiring significantly less time to train. Our model achieves 28.4 BLEU on the WMT 2014 English-to-German translation task, improving over the existing best results, including ensembles, by over 2 BLEU. On the WMT 2014 English-to-French translation task, our model establishes a new single-model state-of-the-art BLEU score of 41.8 after training for 3.5 days on eight GPUs, a small fraction of the training costs of the best models from the literature. We show that the Transformer generalizes well to other tasks by applying it successfully to English constituency parsing both with large and limited training data.

1 Introduction

Recurrent neural networks, long short-term memory [13] and gated recurrent [7] neural networks in particular, have been firmly established as state of the art approaches in sequence modeling and

*Equal contribution. Listing order is random. Jakob proposed replacing RNNs with self-attention and started the effort to evaluate this idea. Ashish, with Illia, designed and implemented the first Transformer models and

Abstract

The dominant sequence transduction models are based on complex recurrent or convolutional neural networks that include an encoder and a decoder. The best performing models also connect the encoder and decoder through an attention mechanism. We propose a new simple network architecture, the Transformer, based solely on attention mechanisms, dispensing with recurrence and convolutions entirely. Experiments on two machine translation tasks show these models to be superior in quality while being more parallelizable and requiring significantly less time to train. Our model achieves 28.4 BLEU on the WMT 2014 English-to-German translation task, improving over the existing best results, including ensembles, by over 2 BLEU. On the WMT 2014 English-to-French translation task, our model establishes a new single-model state-of-the-art BLEU score of 41.8 after training for 3.5 days on eight GPUs, a small fraction of the training costs of the best models from the literature. We show that the Transformer generalizes well to other tasks by applying it successfully to English constituency parsing both with large and limited training data.

Transformer

La naturaleza recurrente de las RNNs no permite el procesamiento paralelo.

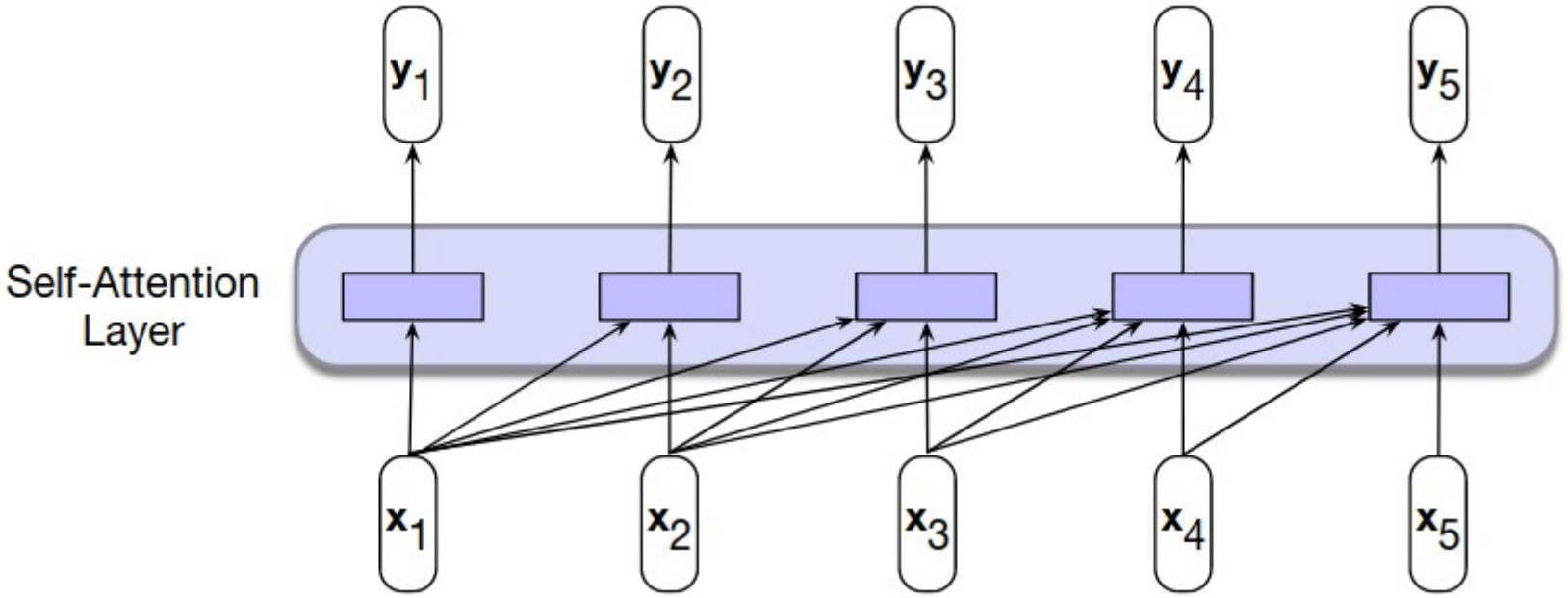
El transformer reúne varias ideas exitosas

- **self-attention**
- positional embeddings
- contextualized embeddings
- fine-tuning
- masked language modeling
- next sentence prediction

El transformer puede procesar todas las entradas en paralelo.

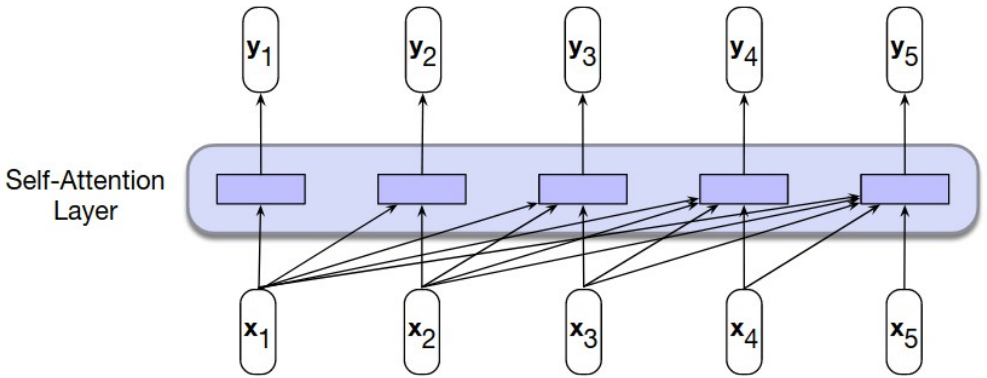
Self-Attention

Transformer: Self-Attention



Cada paso es independiente (se puede paralelizar).

Transformer: Self-Attention



$$\mathbf{y}_i = \sum_{j \le i} \alpha_{ij} \mathbf{x}_j$$

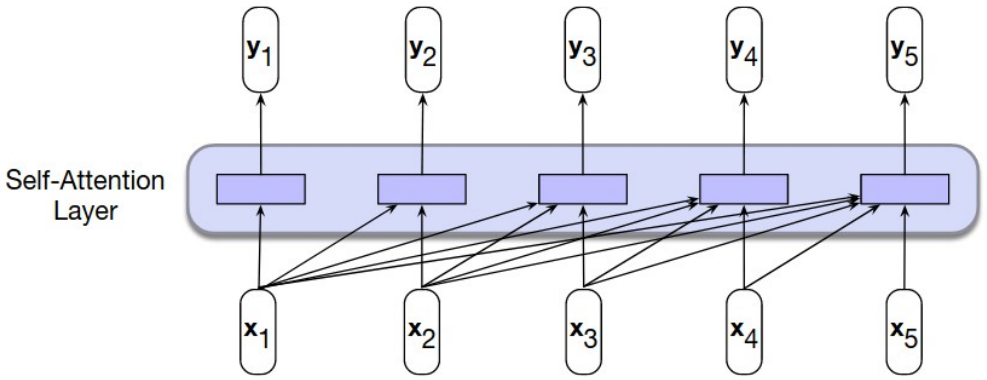
$$\alpha_{ij} = \text{softmax}(\text{score}(\mathbf{x}_i, \mathbf{x}_j)) \;\; \forall j \le i$$

$$= \frac{\exp(\text{score}(\mathbf{x}_i, \mathbf{x}_j))}{\sum_{k=1}^{i} \exp(\text{score}(\mathbf{x}_i, \mathbf{x}_k))} \;\; \forall j \le i$$

$$\text{score}(\mathbf{x}_i, \mathbf{x}_j) = \; ?$$

Basado en https://web.stanford.edu/~jurafsky/slp3/

Transformer: Self-Attention



3 proyecciones de cada $\mathbf{x}_i$ (**query**, **key** y **value**)

$$\mathbf{q}_i = \mathbf{W}^{\mathbf{Q}}\mathbf{x}_i; \;\; \mathbf{k}_i = \mathbf{W}^{\mathbf{K}}\mathbf{x}_i; \;\; \mathbf{v}_i = \mathbf{W}^{\mathbf{V}}\mathbf{x}_i$$

Los vectores $\mathbf{q}_i$ y $\mathbf{k}_j$ computan el **score** de como influye la entrada j para i

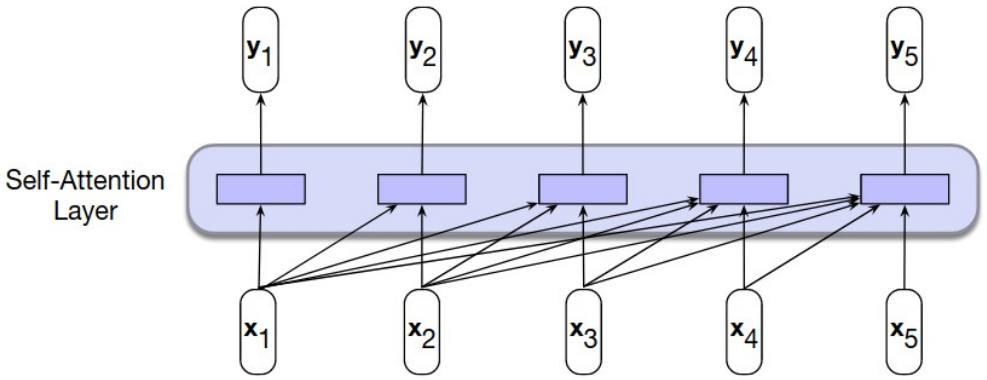
$$\text{score}(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{q}_i \cdot \mathbf{k}_j$$

Luego softmax (en j) y se ponderan los $\mathbf{v}_i$

$$\mathbf{y}_i = \sum_{j \leq i} \alpha_{ij}\mathbf{v}_j$$

Basado en https://web.stanford.edu/~jurafsky/slp3/

Transformer: Self-Attention



3 proyecciones de cada $\mathbf{x}_i$ (**query**, **key** y **value**)

$$\mathbf{q}_i = \mathbf{W}^{\mathbf{Q}}\mathbf{x}_i; \;\; \mathbf{k}_i = \mathbf{W}^{\mathbf{K}}\mathbf{x}_i; \;\; \mathbf{v}_i = \mathbf{W}^{\mathbf{V}}\mathbf{x}_i$$

Los vectores $\mathbf{q}_i$ y $\mathbf{k}_j$ computan el **score** de como influye la entrada j para i

$$\text{score}(\mathbf{x}_i, \mathbf{x}_j) = \frac{\mathbf{q}_i \cdot \mathbf{k}_j}{\sqrt{d_k}}$$ ⇦ (prox. slide)

Luego softmax (en j) y se ponderan los $\mathbf{v}_i$

$$\mathbf{y}_i = \sum_{j \leq i} \alpha_{ij}\mathbf{v}_j$$

Basado en https://web.stanford.edu/~jurafsky/slp3/

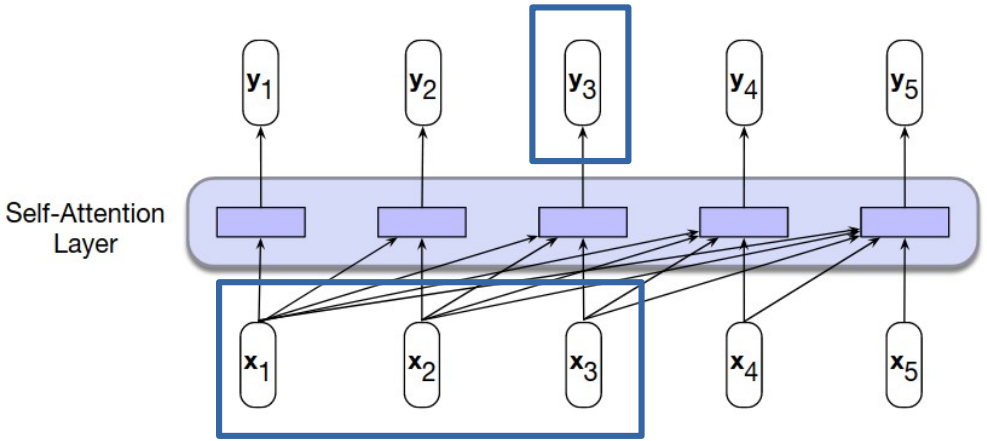
Sobre $\sqrt{d_k}$

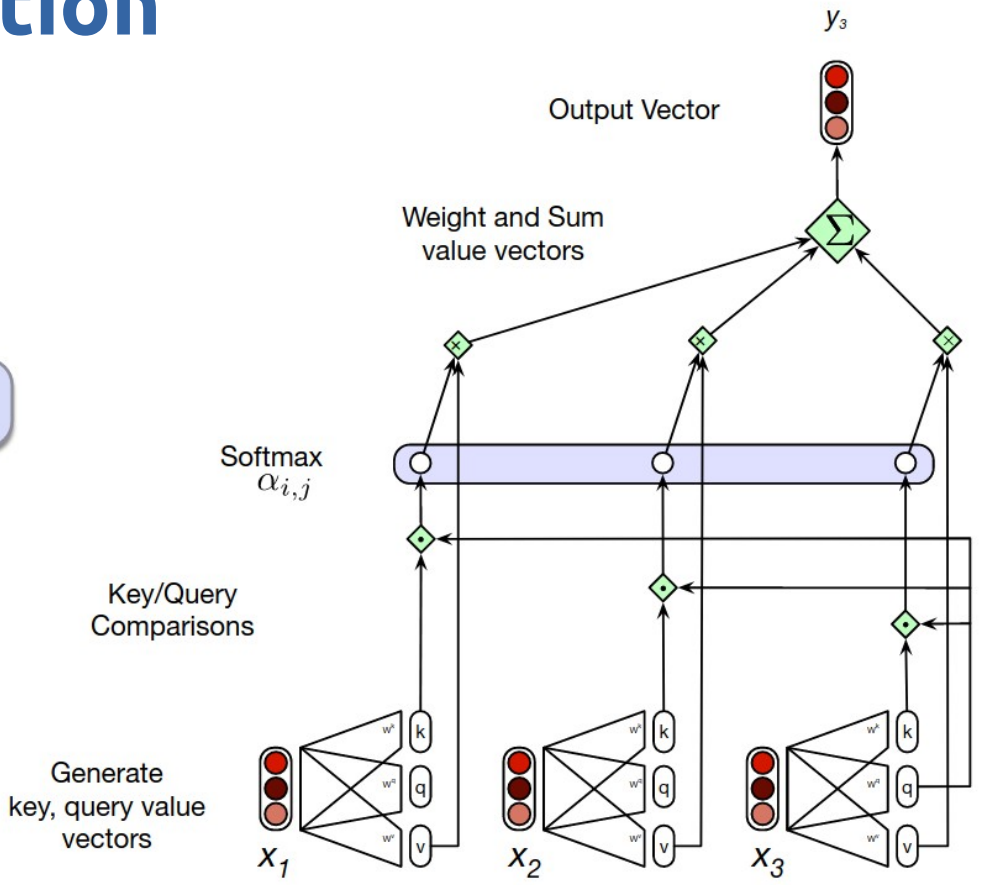
The two most commonly used attention functions are additive attention [2], and dot-product (multiplicative) attention. Dot-product attention is identical to our algorithm, except for the scaling factor of $\frac{1}{\sqrt{d_k}}$. Additive attention computes the compatibility function using a feed-forward network with a single hidden layer. While the two are similar in theoretical complexity, dot-product attention is much faster and more space-efficient in practice, since it can be implemented using highly optimized matrix multiplication code.

While for small values of d_k the two mechanisms perform similarly, additive attention outperforms dot product attention without scaling for larger values of d_k [3]. We suspect that for large values of d_k, the dot products grow large in magnitude, pushing the softmax function into regions where it has extremely small gradients [4]. To counteract this effect, we scale the dot products by $\frac{1}{\sqrt{d_k}}$.

https://arxiv.org/pdf/1706.03762.pdf

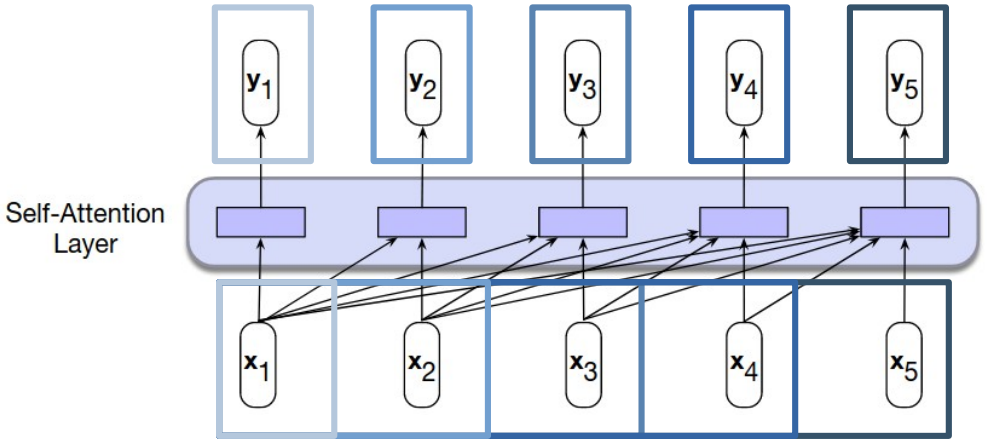
Transformer: Self-Attention





Basado en https://web.stanford.edu/~jurafsky/slp3/

Transformer: Self-Attention



$$\mathbf{q}_i = \mathbf{W}^{\mathbf{Q}}\mathbf{x}_i;\ \ \mathbf{k}_i = \mathbf{W}^{\mathbf{K}}\mathbf{x}_i;\ \ \mathbf{v}_i = \mathbf{W}^{\mathbf{V}}\mathbf{x}_i$$

Paralelizando:

$$\mathbf{Q} = \mathbf{X}\mathbf{W}^{\mathbf{Q}};\ \ \mathbf{K} = \mathbf{X}\mathbf{W}^{\mathbf{K}};\ \ \mathbf{V} = \mathbf{X}\mathbf{W}^{\mathbf{V}}$$

$$\mathit{SelfAttention}(\mathbf{Q},\mathbf{K},\mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^{\mathsf{T}}}{\sqrt{d_k}}\right)\mathbf{V}$$

QK^T ⇨

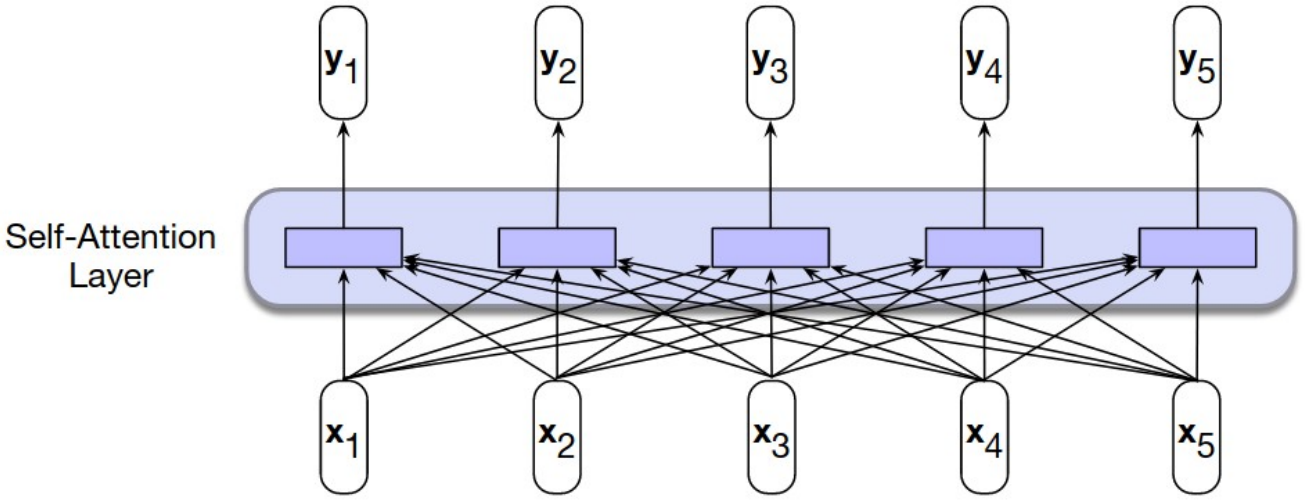
q1•k1	−∞	−∞	−∞	−∞
q2•k1	q2•k2	−∞	−∞	−∞
q3•k1	q3•k2	q3•k3	−∞	−∞
q4•k1	q4•k2	q4•k3	q4•k4	−∞
q5•k1	q5•k2	q5•k3	q5•k4	q5•k5

N (filas) × N (columnas)

Se rellena con -∞ (se vuelven 0 con softmax)

Basado en https://web.stanford.edu/~jurafsky/slp3/

Bidirectional Transformer



N

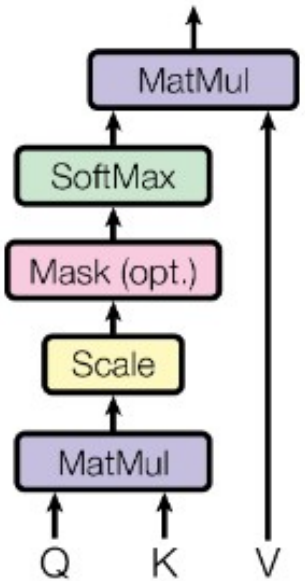
q1•k1	q1•k2	q1•k3	q1•k4	q1•k5
q2•k1	q2•k2	q2•k3	q2•k4	q2•k5
q3•k1	q3•k2	q3•k3	q3•k4	q3•k5
q4•k1	q4•k2	q4•k3	q4•k4	q4•k5
q5•k1	q5•k2	q5•k3	q5•k4	q5•k5

N

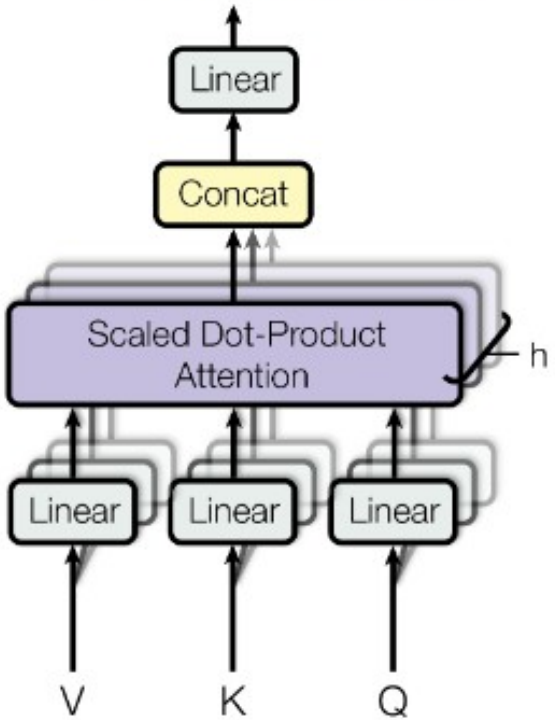
Multihead Self-Attention

Multihead Self-Attention

Scaled Dot-Product Attention



Multi-Head Attention



https://arxiv.org/pdf/1706.03762.pdf

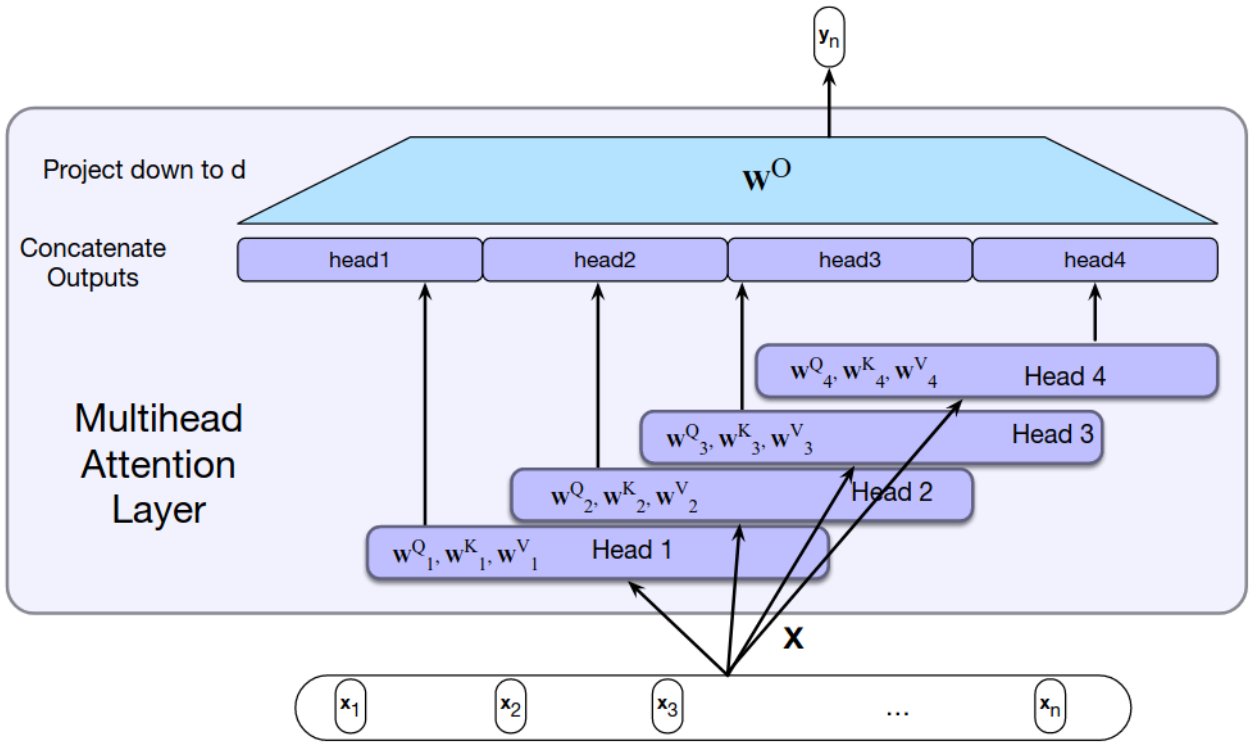
Multihead Self-Attention

$$\mathbf{W}_i^Q \in \mathbb{R}^{d \times d_k},\ \mathbf{W}_i^K \in \mathbb{R}^{d \times d_k},\ \mathbf{W}_i^V \in \mathbb{R}^{d \times d_v}$$

$$MultiHeadAttn(\mathbf{X}) = (\mathbf{head}_1 \oplus \mathbf{head}_2 \ldots \oplus \mathbf{head}_h)\mathbf{W}^O$$

$$\mathbf{Q}_i = \mathbf{X}\mathbf{W}_i^Q\ ;\ \mathbf{K}_i = \mathbf{X}\mathbf{W}_i^K\ ;\ \mathbf{V}_i = \mathbf{X}\mathbf{W}_i^V$$

$$\mathbf{head}_i = SelfAttention(\mathbf{Q}_i, \mathbf{K}_i, \mathbf{V}_i)$$



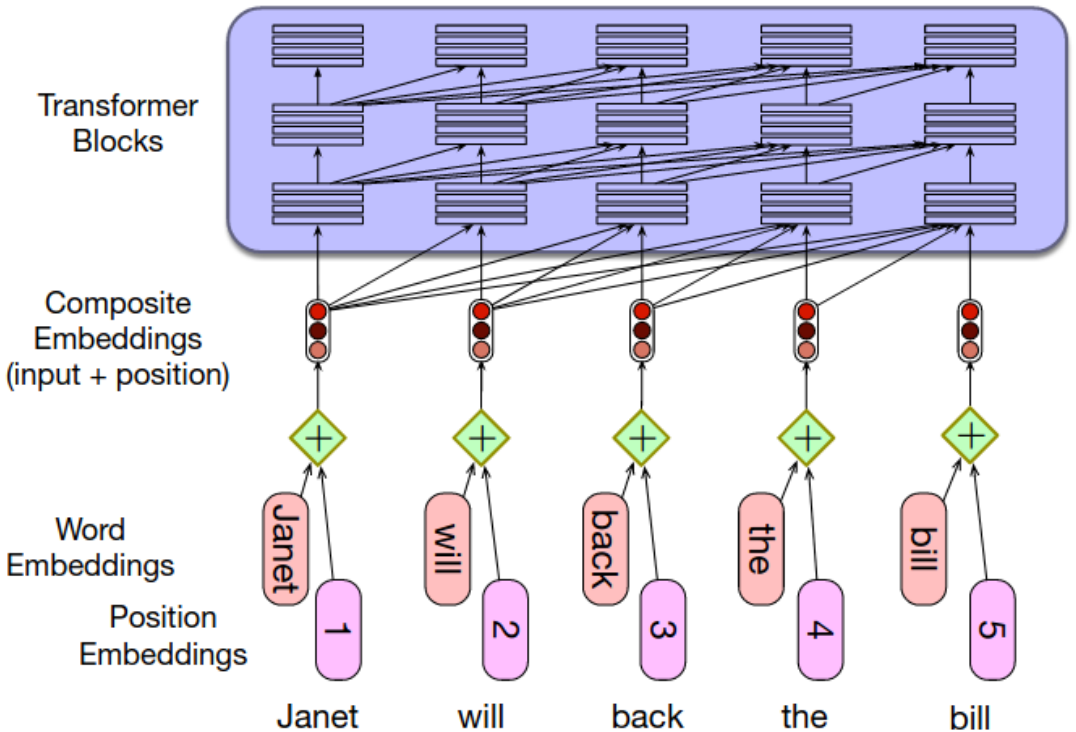
Basado en https://web.stanford.edu/~jurafsky/slp3/

Positional Embeddings

Word Order: Positional Embeddings

Hasta este punto no hay noción de orden en la entrada.

Solucion propuesta: "Aumentar" cada x_i para que represente su posición (**position embeddings)**



Pueden ser entrenados pero las primeras posiciones van a ser más comunes (por los diferentes largos), dificultando el entrenamiento.

Una altarnativa es que sean estáticos. Una combinación de senos y cosenos es usada en el transformer original.

Mejorar los positional embeddings es un área actual de investigación. (mejorar todo el transformer en realidad)

Basado en https://web.stanford.edu/~jurafsky/slp3/

Word Order: Positional Embeddings

Let t be the desired position in an input sentence, $\overrightarrow{p_t} \in \mathbb{R}^d$ be its corresponding encoding, and d be the encoding dimension (where $d \equiv_2 0$) Then $f : \mathbb{N} \to \mathbb{R}^d$ will be the function that produces the output vector $\overrightarrow{p_t}$ and it is defined as follows:

$$\overrightarrow{p_t}^{(i)} = f(t)^{(i)} := \begin{cases} \sin(\omega_k . t), & \text{if } i = 2k \\ \cos(\omega_k . t), & \text{if } i = 2k+1 \end{cases}$$

where

$$\omega_k = \frac{1}{10000^{2k/d}}$$

https://kazemnejad.com/blog/transformer_architecture_positional_encoding/

Word Order: Positional Embeddings

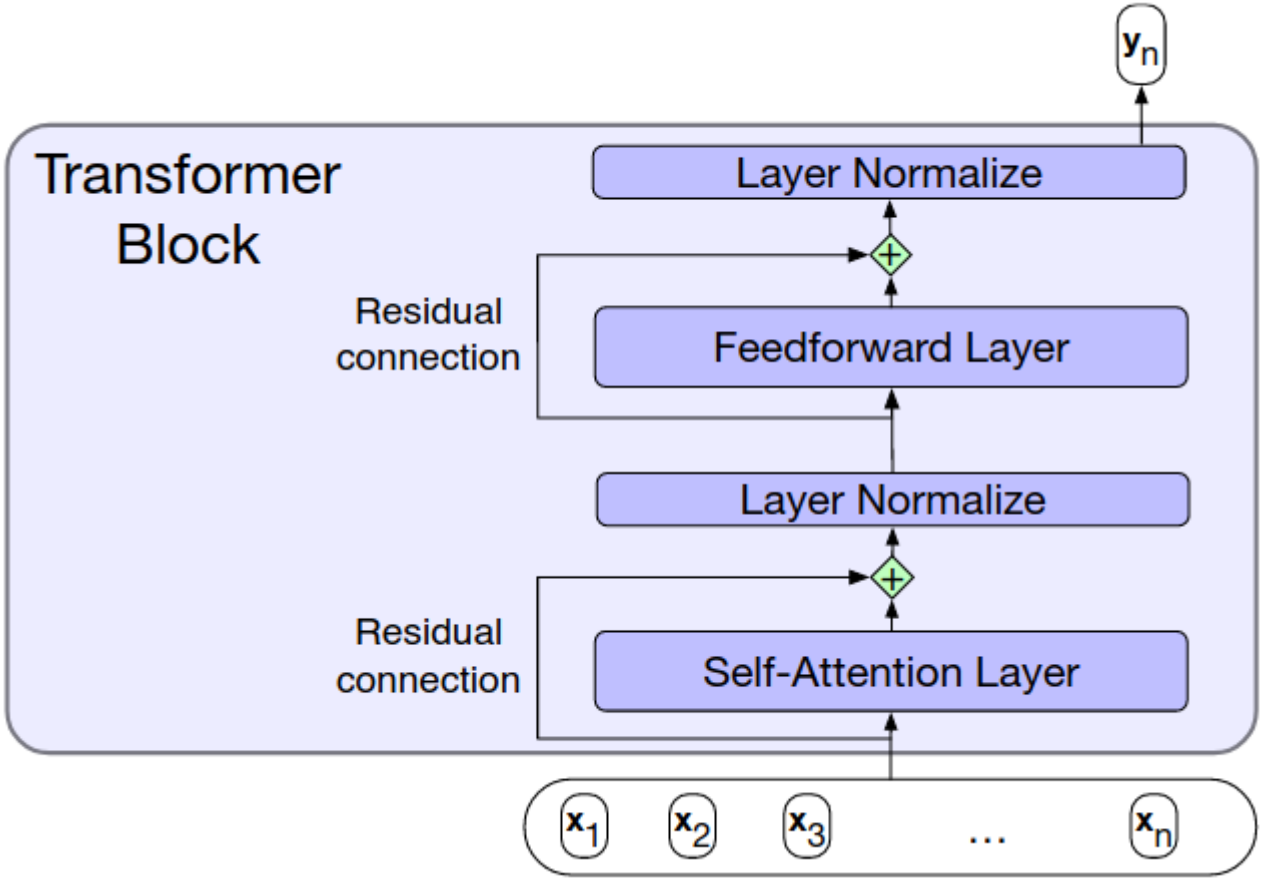
0 :	0 0 0 0	8 :	1 0 0 0
1 :	0 0 0 1	9 :	1 0 0 1
2 :	0 0 1 0	10 :	1 0 1 0
3 :	0 0 1 1	11 :	1 0 1 1
4 :	0 1 0 0	12 :	1 1 0 0
5 :	0 1 0 1	13 :	1 1 0 1
6 :	0 1 1 0	14 :	1 1 1 0
7 :	0 1 1 1	15 :	1 1 1 1

En los distintas componentes de una representación binaria se pueden observar frecuencias de cambio. Utilizar valores bianarios desaprovecharía el espacio, en su lugar se hace una variante continua (con senos y cosenos).

https://kazemnejad.com/blog/transformer_architecture_positional_encoding/

Transformer Block

Transformer Block



$$\mathbf{z} = \text{LayerNorm}(\mathbf{x} + \text{SelfAttn}(\mathbf{x}))$$
$$\mathbf{y} = \text{LayerNorm}(\mathbf{z} + \text{FFNN}(\mathbf{z}))$$

Layer Normalization

$$\mu = \frac{1}{d_h}\sum_{i=1}^{d_h} x_i$$

$$\sigma = \sqrt{\frac{1}{d_h}\sum_{i=1}^{d_h}(x_i - \mu)^2}$$

$$\hat{\mathbf{x}} = \frac{(\mathbf{x} - \mu)}{\sigma}$$

(γ y β se aprenden)

$$LayerNorm = \gamma\,\hat{\mathbf{x}} + \beta$$

https://web.stanford.edu/~jurafsky/slp3/

Transformer Network

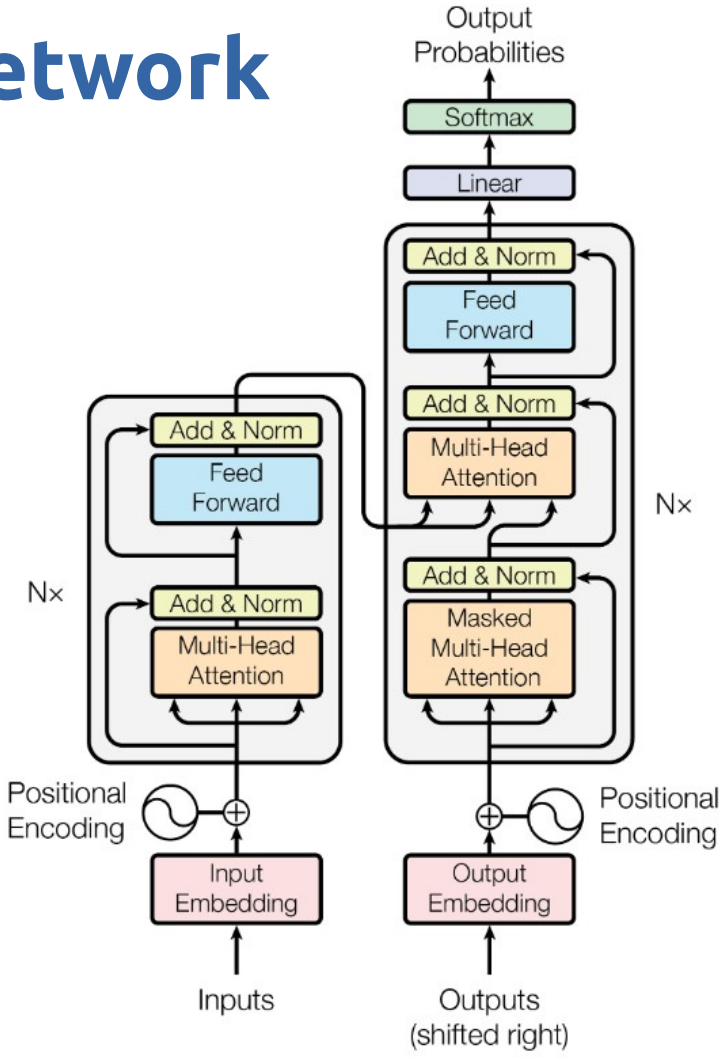
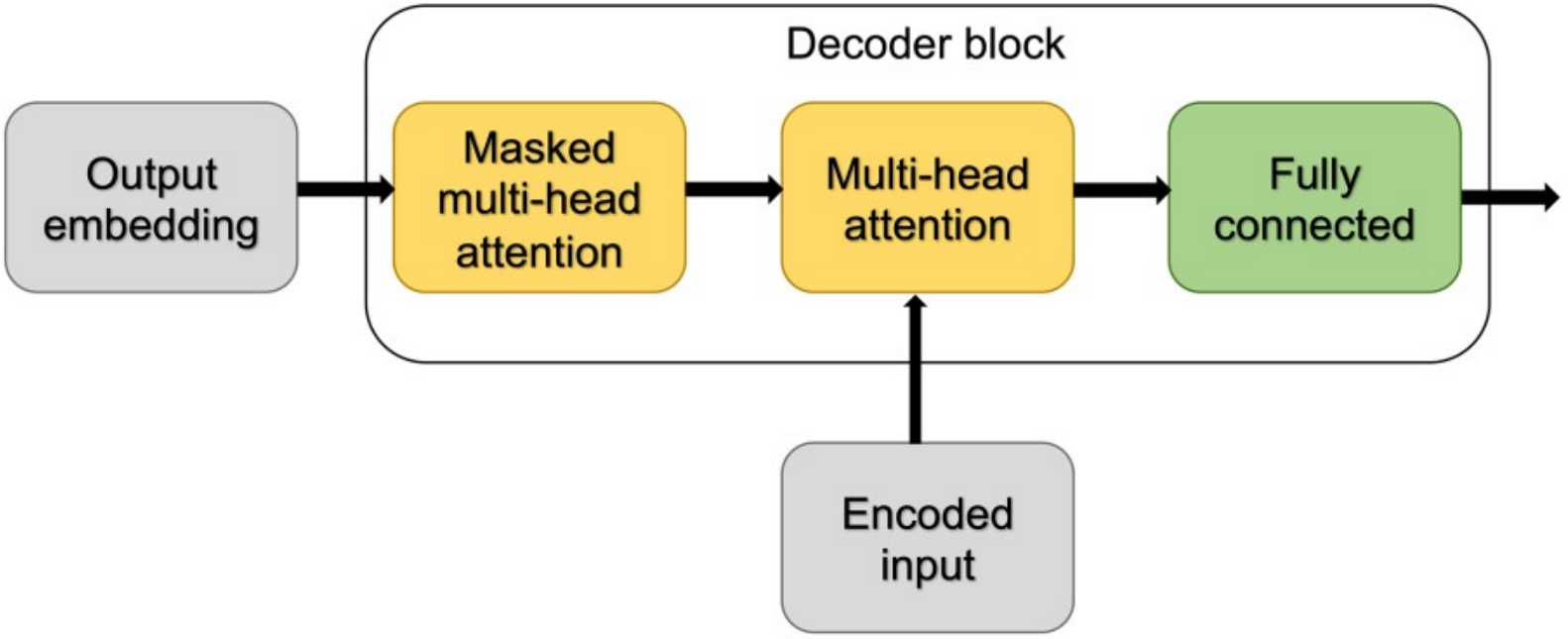


Figure 1: The Transformer - model architecture.

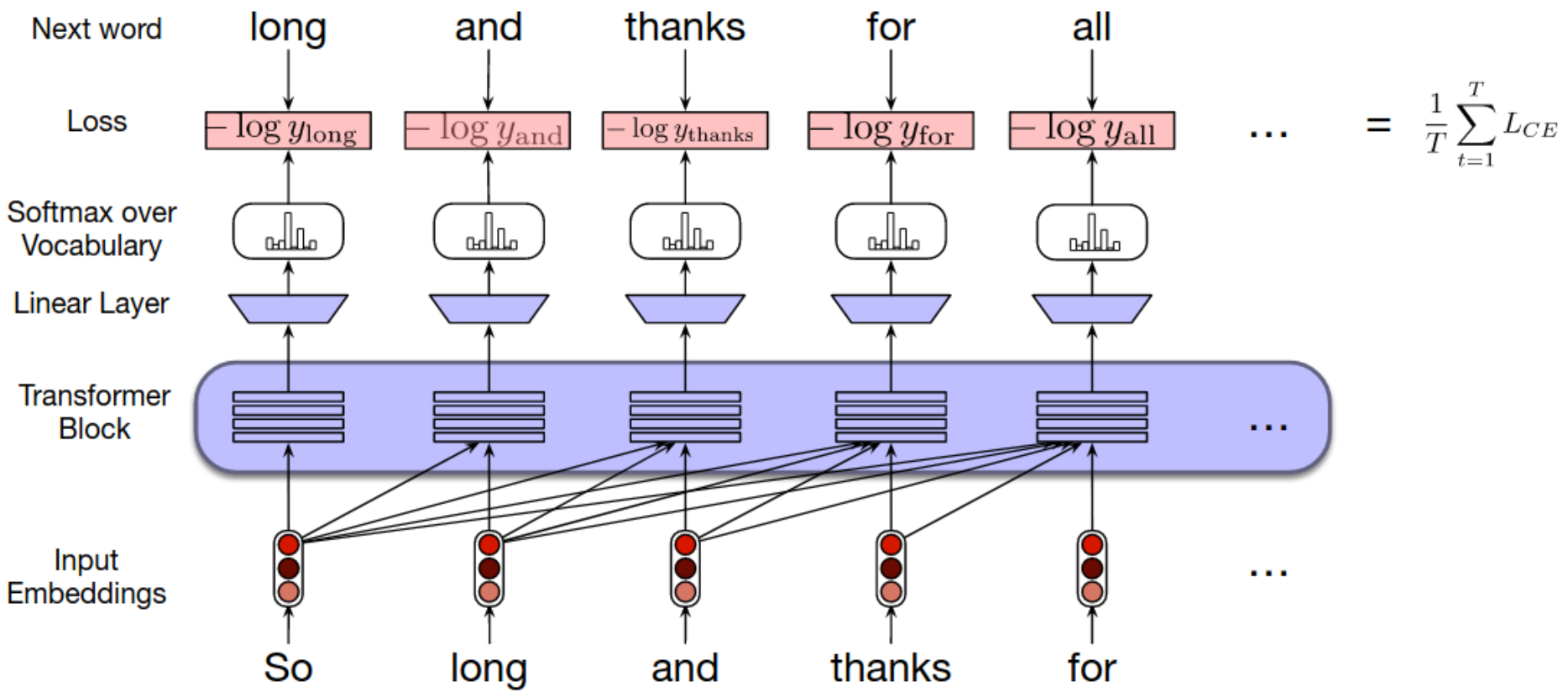
https://arxiv.org/pdf/1706.03762.pdf

Decoder block

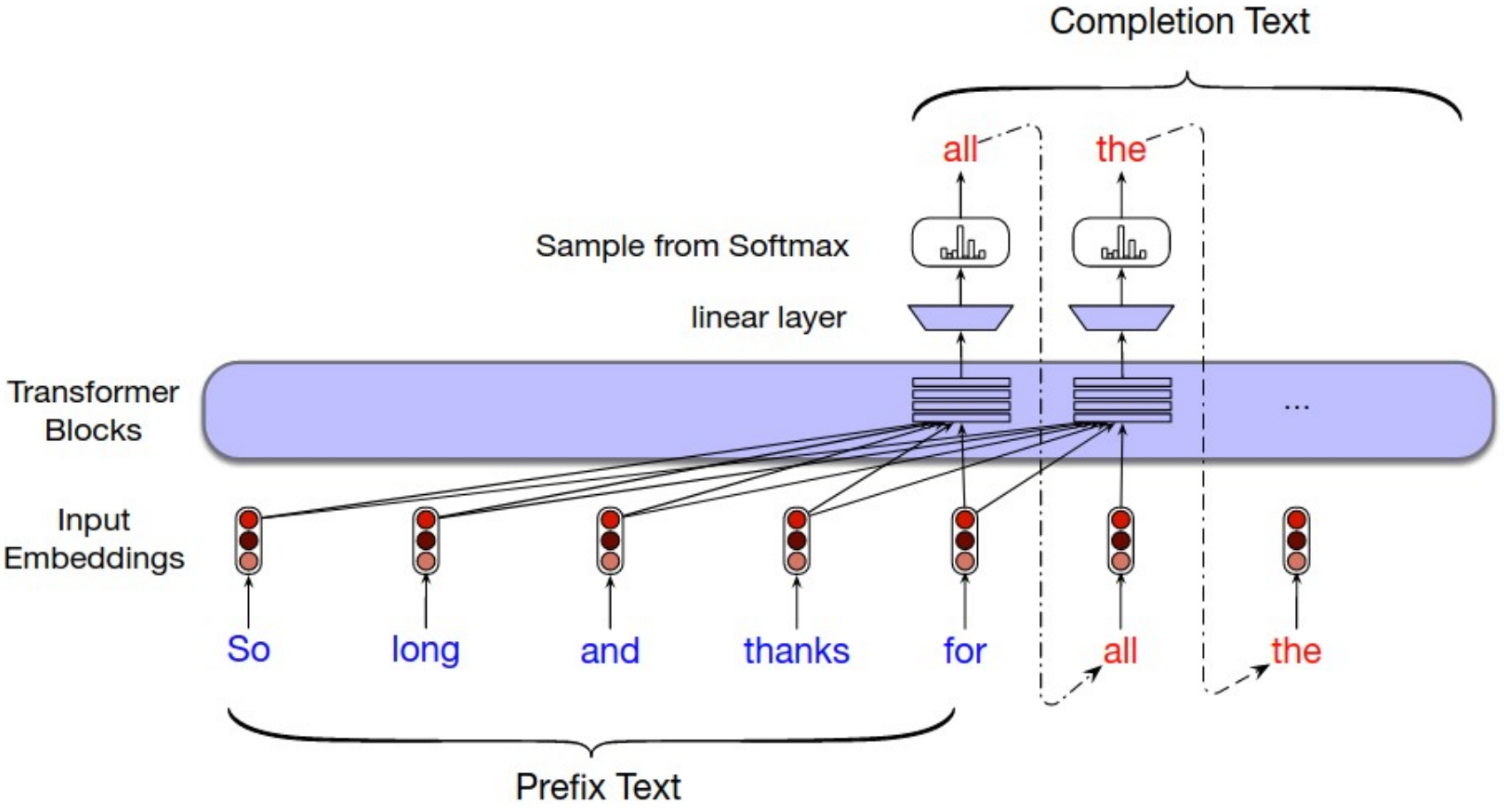


¿Cómo usamos al Transformer?

Modelo de Lenguaje (Next Word Prediction)

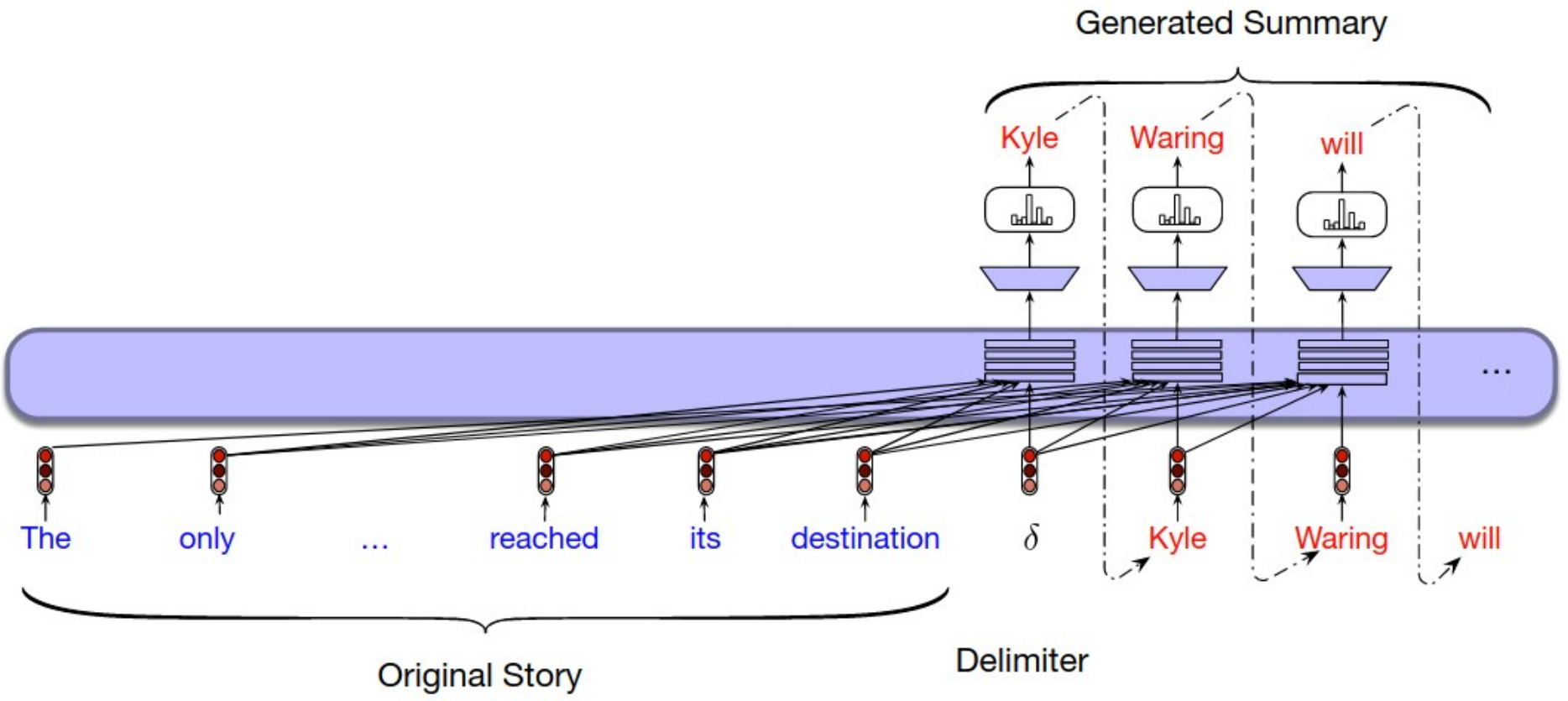


Generar con Transformers



https://web.stanford.edu/~jurafsky/slp3/

Generación Condicionada



Basado en https://web.stanford.edu/~jurafsky/slp3/

Pretraining y fine-tuning

Pretraining
Se preentrena un transformer (ej. para a predecir la siguiente palabra) en un corpus grande.

Fine-tuning
En modelo **preentrendado**, se agrega una capa feedforward que es ajustada con un corpus anotado para una tarea específica.

Improving Language Understanding by Generative Pre-Training

Alec Radford
OpenAI
alec@openai.com

Karthik Narasimhan
OpenAI
karthikn@openai.com

Tim Salimans
OpenAI
tim@openai.com

Ilya Sutskever
OpenAI
ilyasu@openai.com

Abstract

Natural language understanding comprises a wide range of diverse tasks such as textual entailment, question answering, semantic similarity assessment, and document classification. Although large unlabeled text corpora are abundant, labeled data for learning these specific tasks is scarce, making it challenging for discriminatively trained models to perform adequately. We demonstrate that large gains on these tasks can be realized by *generative pre-training* of a language model on a diverse corpus of unlabeled text, followed by *discriminative fine-tuning* on each specific task. In contrast to previous approaches, we make use of task-aware input transformations during fine-tuning to achieve effective transfer while requiring minimal changes to the model architecture. We demonstrate the effectiveness of our approach on a wide range of benchmarks for natural language understanding. Our general task-agnostic model outperforms discriminatively trained models that use architectures specifically crafted for each task, significantly improving upon the state of the art in 9 out of the 12 tasks studied. For instance, we achieve absolute improvements of 8.9% on commonsense reasoning (Stories Cloze Test), 5.7% on question answering (RACE), and 1.5% on textual entailment (MultiNLI).

2018