Clase 142 — Mecanismos de atención

Parte: 2 — Deep Learning · Fuente: Géron, cap. 16 § Attention Mechanisms + Bahdanau et al. (2015), Luong et al. (2015). Duración estimada: 75 min.

Clase 142 — Mecanismos de atención

> *Parte: 2 — Deep Learning · Fuente: Géron, cap. 16 § Attention Mechanisms + Bahdanau et al. (2015), Luong et al. (2015). Duración estimada: 75 min.*

Objetivo

Entender la atención — el mecanismo que destrabó NLP moderno. Bahdanau (2015): permite al decoder mirar todos los hidden states del encoder, ponderando dinámicamente. Luego self-attention (Vaswani 2017): tokens dentro de la misma secuencia se atienden entre sí → Transformer (clase 126).

Resultados de aprendizaje

Al finalizar, el estudiante podrá:

- Explicar scaled dot-product attention: softmax(QK^T / √d) V.
- Diferenciar cross-attention (decoder→encoder, clásico Bahdanau) de self-attention (intra-secuencia, base del Transformer).
- Implementar attention a mano en numpy/TF para una secuencia corta.
- Aplicar keras.layers.Attention o keras.layers.MultiHeadAttention.
- Interpretar attention weights como "qué tokens fuente miró el decoder al generar cada token destino".

Temas

- Motivación: bottleneck del encoder en seq2seq.
- Bahdanau (additive) vs Luong (multiplicative / dot-product).
- Q, K, V: queries, keys, values.
- Scaled dot-product attention.
- Self-attention vs cross-attention.
- Multi-head: paralelizar varias attention heads con subspaces distintos.
- Attention weights visualizables.

Definiciones y características

- Query (Q): "lo que estoy buscando" (e.g., el token del decoder).
- Key (K): "lo que cada elemento ofrece como índice" (e.g., los hidden del encoder).
- Value (V): "el contenido a recuperar" (normalmente igual o relacionado a K).
- Attention weights: softmax(QK^T / √d), matrix (seq_len_q, seq_len_k).
- Scaled: dividir por √d para que el softmax no sature.
- Multi-head: dividir d en h heads, calcular attention por separado, concatenar.

Dataset / recursos

- Reusar dataset de traducción de 124.
- Librerías: tensorflow, keras.

Ejercicios

1. Attention a mano: Q, K, V random (seq, d); calcular softmax(QK^T / √d) V. Verificar shapes.
2. Visualizar attention map: tras entrenar un seq2seq con atención, plot heatmap de los pesos (target, source) para una traducción.
3. MultiHeadAttention: mha = MultiHeadAttention(num_heads=8, key_dim=64); output = mha(query, value).
4. Self-attention: aplicar mha(x, x) (query = key = value). Esto es el bloque de Transformer.
5. Causal mask: para generación autoregresiva, MultiHeadAttention(..., use_causal_mask=True).

Homework verificable

Traducción con atención sobre el dataset de 124:

1. Encoder LSTM con return_sequences=True.
2. Decoder con cross-attention sobre todos los outputs del encoder.
3. Train y evaluar BLEU.
4. Visualizar attention map para 2 frases.

Criterio de aceptación: BLEU debe mejorar respecto al seq2seq sin atención (de 124) por ≥ 5 pp; el attention map muestra alineamiento source-target razonable.

Errores comunes

Síntoma / mensaje	Causa y cómo arreglar
Olvido scale / √d	Softmax satura con d grande. Fix: usar Mul
Attention sin masking sobre padding	Atiende a tokens <pad>. Fix: attention_mas
Causal mask al revés	Mira el futuro. Fix: use_causal_mask=True
Cross-attention con Q de un tamaño y K/V d	OK si d_q != d_kv, pero MultiHeadAttention
Visualizar attention de un modelo no conve	Mapas ruidosos. Fix: entrenar bien primero

Preguntas frecuentes

¿Self vs cross attention?

Self: Q = K = V (la secuencia se atiende a sí misma). Cross: Q de un origen, K/V de otro (decoder mira encoder).

¿Por qué √d y no d en el scaled?

QK^T tiene varianza ~ d. Dividir por √d mantiene varianza estable y softmax no satura.

¿Multi-head qué aporta?

Cada head puede aprender un patrón distinto (one por sintaxis, otro por semántica, ...). Empíricamente importante.

¿Attention O(N²)?

Sí, en memoria y compute. Es el principal limitante para secuencias largas. Flash Attention (clase 126) lo optimiza; Linformer/Performer aproximan.

¿Atención biológica?

Inspiración loose. El mecanismo no es realmente cómo funciona el cerebro, pero la metáfora "prestar atención a partes relevantes" es intuitiva.

Referencias

- Géron, cap. 16 — Attention Mechanisms.
- Bahdanau, Cho & Bengio (2015), Neural Machine Translation by Jointly Learning to Align and Translate, ICLR.
- Luong, Pham & Manning (2015), Effective Approaches to Attention-based Neural Machine Translation, EMNLP.
- Vaswani et al. (2017), Attention Is All You Need, NeurIPS — paper del Transformer.

Siguiente clase

Clase 143 — Transformers: arquitectura, BERT, GPT (+ Flash Attention, RoPE, GQA)

Apéndice: notebook (primer bloque)

Primera celda ejecutable del notebook de la clase.

```
# Imports y configuración inicial
```

Archivos complementarios

- notebook.ipynb