1a Emisión

Módulo 8

Introducción al Deep Learning

Modelos basados en atención

Instructor: Gibran Fuentes Pineda

DGTIC

UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO
Dirección General de Cómputo y de Tecnologías de información y Comunicación
Dirección de Docencia en TIC

Educación Continua
1971 - 2021

Objetivo del tema

- Identificar los componentes y el funcionamiento de las redes neuronales basadas en atención, así como estrategias para entrenarlas.

Sub-temas

8.1 Motivación
8.2 Mecanismos de atención
8.3 Autoatención
8.4 Atención multi-cabeza
8.5 Bloq y arquitecturas tipo transformer
8.6 Ejemplos prácticos: generación de texto y detección de polaridad en texto

Motivación: modelos secuencia a secuencia (seq2seq)

- Necesitan codificar todo el contexto de la entrada en un sólo vector

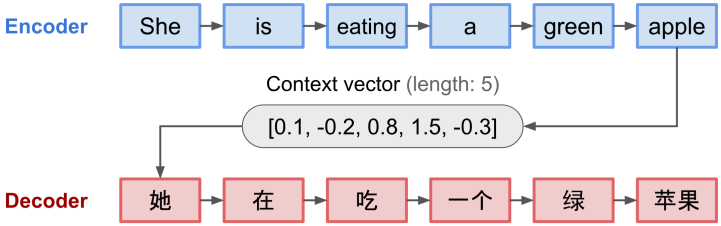

Imagen tomada de https://lilianweng.github.io/lil-log/2018/06/24/attention-attention.html

Mecanismos de atención: intuición

- Información relevante de la entrada se codifica en un solo vector de contexto (último estado de una red recurrente)
  - Difícil en secuencias largas
- Mecanismos de atención: se calcula un vector de contexto distinto por cada paso del decodificador a partir de:
  1. Estados del decodificador
  2. Estados del codificador
  3. Función de alineación

Mecanismos de atención: esquema general

- Dado un paso t del decodificador, se calcula un vector de contexto $\mathbf{c}^{[t]}$ a partir de todos los estados del codificador

$$\mathbf{c}^{[t]} = \sum_{i=1}^{T} \alpha_{t,i} \cdot \widehat{\mathbf{h}}^{[i]}$$

donde $\widehat{\mathbf{h}}^{[i]}$ es el estado del codificador en el paso i, T es el número total de pasos del codificador y $\alpha_{t,i} = \mathit{softmax}(\mathit{puntaje}(\mathbf{h}^{[t]}, \widehat{\mathbf{h}}^{[i]}))$

- *puntaje* es una función que mide la importancia de 2 estados y $\mathbf{h}^{[t]}$ es el estado en el paso t del decodificador.

Mecanismos de atención: funciones *puntaje*

- Basadas en contenido

$$\text{puntaje}(\mathbf{h}^{[t]}, \hat{\mathbf{h}}^{[i]}) = \mathbf{h}^{[t]\top}\hat{\mathbf{h}}^{[i]} \text{ (producto punto)}$$

$$\text{puntaje}(\mathbf{h}^{[t]}, \hat{\mathbf{h}}^{[i]}) = \frac{\mathbf{h}^{[t]\top}\hat{\mathbf{h}}^{[i]}}{\sqrt{T}} \text{ (producto punto escalado)}$$

$$\text{puntaje}(\mathbf{h}^{[t]}, \hat{\mathbf{h}}^{[i]}) = \mathbf{h}^{[t]\top}\mathbf{W}_a\hat{\mathbf{h}}^{[i]} \text{ (general)}$$

$$\text{puntaje}(\mathbf{h}^{[t]}, \hat{\mathbf{h}}^{[i]}) = \mathbf{W}_a\left[\mathbf{h}^{[t]}; \hat{\mathbf{h}}^{[i]}\right] \text{ (concatenación)}$$

- Basada en ubicación

$$\alpha_{t,i} = \mathbf{W}_a\mathbf{h}^{[t]}$$

Mecanismos de atención: Bahdanau

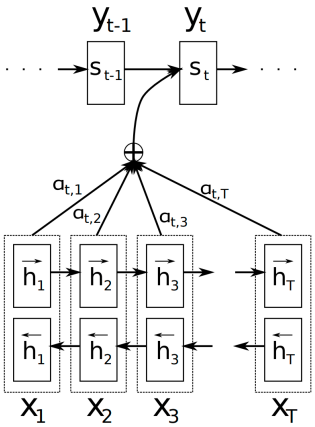

Imagen tomada de Bahdanau et al. Neural machine translation by jointly learning to align and translate, ICLR, 2015

Mecanismos de atención: Luong (global)

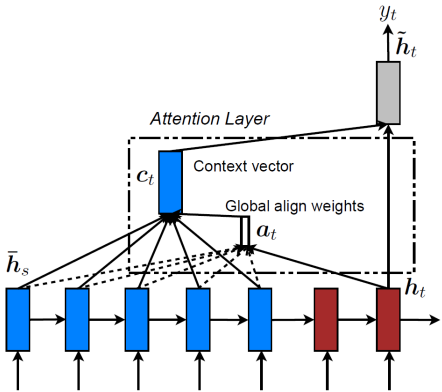

Imagen tomada de Luong et al. Effective approaches to attention-based neural machine translation, EMNLP, 2015

Mecanismos de atención: Luong (local)

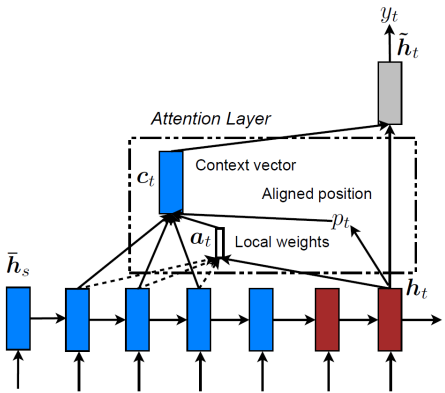

Imagen tomada de Luong et al. Effective approaches to attention-based neural machine translation, EMNLP, 2015

Mecanismos de atención: Luong (alimentación de entradas)

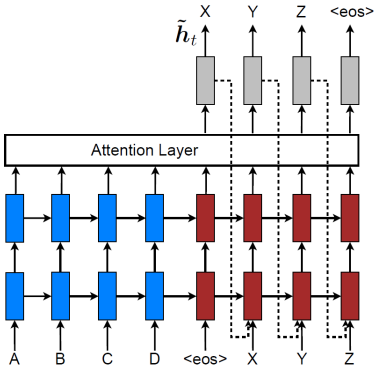

Imagen tomada de Luong et al. Effective approaches to attention-based neural machine translation, EMNLP, 2015

Mecanismos de atención: visualización

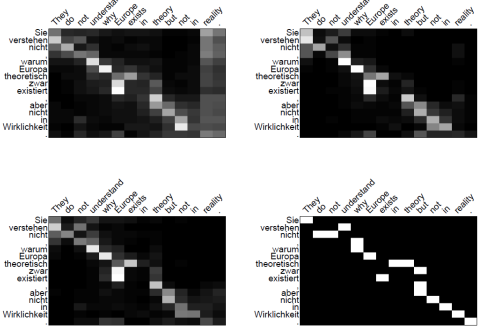

Imagen tomada de Bahdanau et al. Neural machine translation by jointly learning to align and translate, ICLR, 2015

Mecanismos de atención: descripción de imágenes (1)

- Permite enforcarse a sólo ciertas partes de la imagen entrada al producir cada palabra en la salida

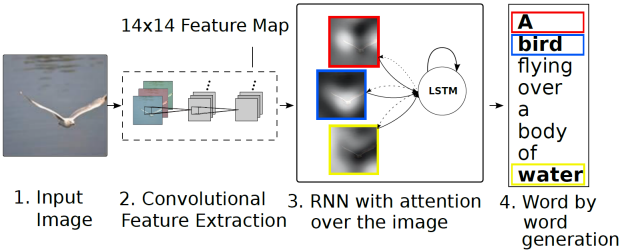

Imagen tomada de Xu et al. Show, Attend and Tell: Neural Image Caption Generation with Visual Attention, ICML, 2015

Mecanismos de atención: descripción de imágenes (2)

- **Dura:** se toma en cuenta una sola region de la imagen
- **Suave:** se toma en cuenta cada región de la imagen en distinta proporción, de forma similar a la atención de Bahdanau

Figure 2. Attention over time. As the model generates each word, its attention changes to reflect the relevant parts of the image. "soft" (top row) vs "hard" (bottom row) attention. (Note that both models generated the same captions in this example.)

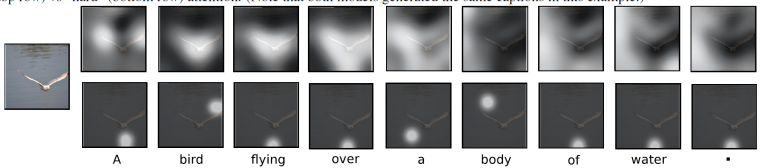

Imagen tomada de Xu et al. Show, Attend and Tell: Neural Image Caption Generation with Visual Attention, ICML, 2015

Autoatención[1]

- Cada salida $\mathbf{y}^{[i]}$ es simplemente la suma ponderada de todas las entradas $\left(\mathbf{x}^{[1]}, \mathbf{x}^{[2]}, \ldots, \mathbf{x}^{[T]}\right)$ en la secuencia:

$$\mathbf{y}^{[i]} = \sum_{j=1}^{T} \alpha_{i,j} \cdot \mathbf{x}^{[j]}, \text{ donde } \sum_{j=1}^{T} \alpha_{i,j} = 1$$

- Cada valor de atención $\alpha_{i,j}$ se obtiene a partir de una función de la entrada $\mathbf{x}^{[j]}$ correspondiente a la salida $\mathbf{y}^{[i]}$ y cada entrada $\mathbf{x}^{[j]}$.

- Una función comúnmente utilizada es

$$\alpha_{i,j} = softmax(\mathbf{x}^{[i]\top} \mathbf{x}^{[j]})$$

[1] Nota que la autoatención no considera el orden.

Bloques y arquitecturas tipo Transformer

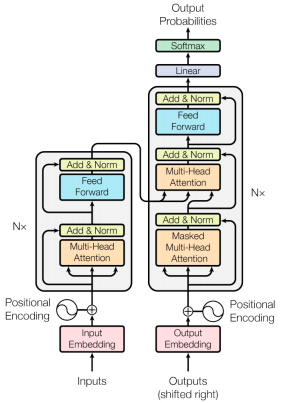

Imagen tomada de Vaswani et al. Attention Is All You Need, NIPS, 2017

Bloques y arquitecturas tipo transformer: autoatención (1)

- Se transforma linealmente cada entrada $\mathbf{x}^{[i]}$ a los vectores consulta $\mathbf{q}^{[i]}$, llave $\mathbf{k}^{[i]}$ y valor $\mathbf{v}^{[i]}$

$$\mathbf{q}^{[i]} = \mathbf{W}_q \mathbf{x}^{[i]}$$
$$\mathbf{k}^{[i]} = \mathbf{W}_k \mathbf{x}^{[i]}$$
$$\mathbf{v}^{[i]} = \mathbf{W}_v \mathbf{x}^{[i]}$$

- La salida en el paso *i* se obtiene de la siguiente manera

$$\mathbf{y}^{[i]} = \sum_{j=1}^{T} \alpha_{i,j} \cdot \mathbf{v}^{[j]}$$

Bloques y arquitecturas tipo transformer: autoatención (2)

$$\mathbf{q}^{[i]} = \mathbf{W}_q \mathbf{x}^{[i]}$$
$$\mathbf{k}^{[i]} = \mathbf{W}_k \mathbf{x}^{[i]}$$
$$\mathbf{v}^{[i]} = \mathbf{W}_v \mathbf{x}^{[i]}$$
$$\mathbf{y}^{[i]} = \sum_{j=1}^{T} \alpha_{i,j} \cdot \mathbf{v}^{[j]}$$

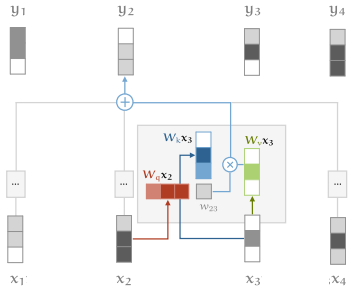

Imagen tomada de http://www.peterbloem.nl/blog/transformers

Transformer: producto punto normalizado

- Considerando que $\mathbf{q}^{[i]}, \mathbf{k}^{[i]} \in \mathbb{R}^{d_k}$ y $\mathbf{v}^{[i]} \in \mathbb{R}^{d_v}$, la función de puntaje está dada por el producto punto normalizado

$$puntaje(\mathbf{x}^{[i]}, \mathbf{x}_j) = \frac{\mathbf{q}^{[i]\top}\mathbf{k}^{[i]}}{\sqrt{d_k}}$$

- Por lo tanto, cada valor de atención $\alpha_{i,j}$ estaría dado por

$$\alpha_{i,j} = softmax\left(\frac{\mathbf{q}^{[i]\top}\mathbf{k}^{[i]}}{\sqrt{d_k}}\right)$$

Bloques y arquitecturas tipo transformer: múltiples cabezas

- Se transforma cada entrada con h distintas $\mathbf{W}_q$, $\mathbf{W}_k$, $\mathbf{W}_v$ y se calcula la salida de la autoatención con cada una
- Se concatenan las salidas de la autoatención de cada entrada y se multiplican por $\mathbf{W}_o$

$$\text{MultiHead}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \underbrace{[\mathbf{y}_1, \ldots, \mathbf{y}_h]}_{\text{Concatenación}} \cdot \mathbf{W}_o$$

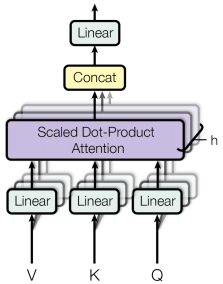

Imagen tomada de Vaswani et al. Attention Is All You Need, NIPS, 2017

Bloques y arquitecturas tipo transformer: codificación posicional

- Para tomar en cuenta el orden en una secuencia, se codifica la posición de cada entrada[2]

$$PE(pos, 2i) = sin\left[\frac{pos}{10000^{\left(\frac{2i}{d_{model}}\right)}}\right]$$

$$PE(pos, 2i+1) = cos\left[\frac{pos}{10000^{\left(\frac{2i}{d_{model}}\right)}}\right]$$

donde *pos* es la posición en la secuencia, *i* la dimensión del *embedding* de entrada y d_{model} su tamaño.

[2] También es posible aprender la codificación. Por ej. Gehring et al. *Convolutional Sequence to Sequence Learning*, arxiv:1705.03122, 2017.

Bloques y arquitecturas tipo transformer: red hacia adelante por posición

- Las salidas de los bloques de autoatención se conectan a 2 capas densas, la primera con función de activación ReLU

$$FFN(X) = \text{máx}(0, x \cdot \mathbf{W}^{\{1\}} + b^{\{1\}}) \cdot \mathbf{W}^{\{2\}} + b^{\{2\}}$$

- Esto se realiza de forma separada por cada entrada en la secuencia[3]

- Llamada red hacia adelante por posición o *Position-wise Feed-Forward Networks*

[3] Esto se puede ver como una convolución 1D con un filtro de tamaño 1.

Bloques y arquitecturas tipo transformer: diseño y construcción

- Los bloques Transformer están compuestos de una capa de autoatención multicabeza seguida de una red hacia adelante por posición, con conexiones residuales y normalización por capa (*Layer Normalization*) en ambos

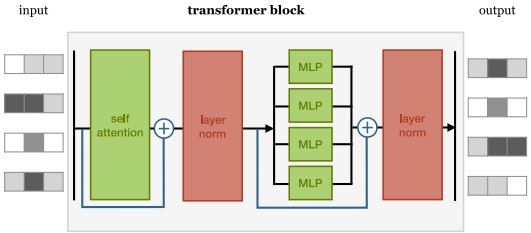

Imagen tomada de Peter Bloem. Transformers from scratch, 2019.

Ejemplo práctico: generación de texto (1)

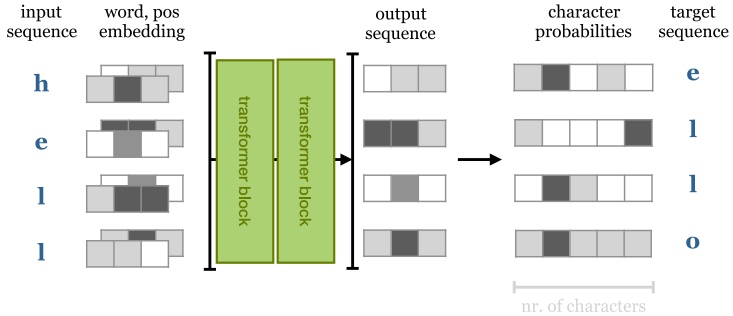

Imagen tomada de Peter Bloem. Transformers from scratch, 2019.

Ejemplo práctico: generación de texto (2)

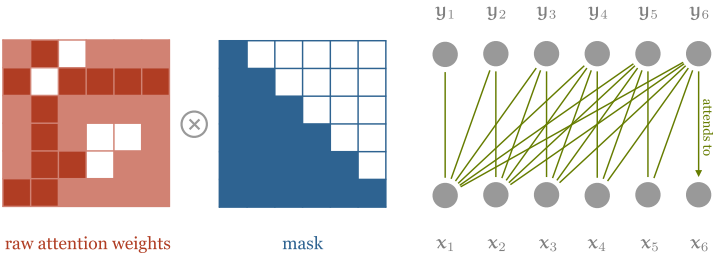

Imagen tomada de Peter Bloem. Transformers from scratch, 2019.

Ejemplo práctico: detección de polaridad en texto

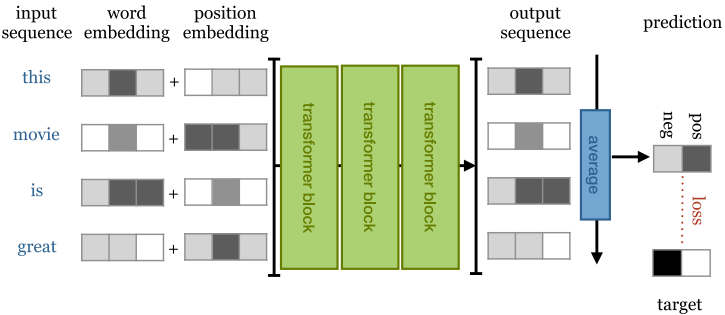

Imagen tomada de Peter Bloem. Transformers from scratch, 2019.

Modelos de lenguaje y aprendizaje por transferencia: definición

- Un modelo de lenguaje es una distribución de probabilidad de una secuencia de T palabras

$$P(x^{[1]}, \ldots, x^{[T]}) = P(x^{[1]}) \cdot \prod_{i=2}^{T} P(x^{[i]} | x^{[1]}, \ldots, x^{[i-1]})$$

- Redes autoregresivas tipo Transformer se han usado como modelos del lenguaje
  - Usualmente entrenadas con tareas de aprendizaje no supervisado
  - Muy efectivas para el aprendizaje por transferencia

Modelos de lenguaje y aprendizaje por transferencia: BERT

- Codificador bidireccional Transformer (sin decodificador)
  - Entrenado con dos tareas: modelado del lenguaje enmascarado y predicción de la siguiente oración
  - Una vez entrenado, se puede emplear para otras tareas realizando un ajuste fino supervisado

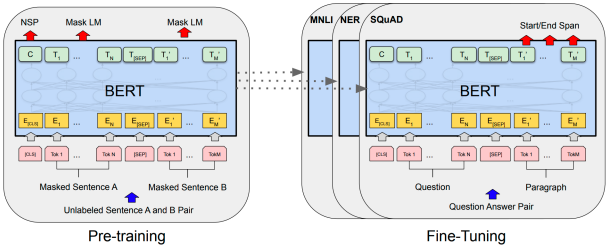

Imagen tomada de Devlin et al. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, NAACL, 2019.

Modelos de lenguaje y aprendizaje por transferencia: parámetros

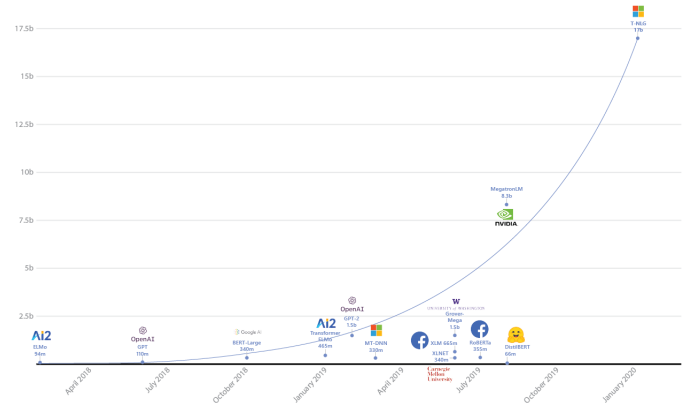

Imagen tomada de https://www.microsoft.com/en-us/research/blog/turing-nlg-a-17-billion-parameter-language-model-by-microsoft/.

Referencias

- Zhang, A., Lipton, Z. C., Li, M., & Smola, A. J. (2021). Dive into deep learning. Capítulo 10. Disponible en http://www.d2l.ai/.
- Murphy, K. P. (2022). Probabilistic Machine Learning: An introduction. MIT Press. Capítulo 15, secciones 15.4–15.7. Disponible en https://probml.github.io/pml-book/book1.html.
- Bloem, P. (2019). Transformers from scratch. Disponible en http://peterbloem.nl/blog/transformers.
- Olah, C., & Carter, S. (2016). Attention and Augmented Recurrent Neural Networks. Disponible en https://distill.pub/2016/augmented-rnns/.
- Weng, L. (2018). Attention? Attention. Disponible en https://lilianweng.github.io/lil-log/2018/06/24/attention-attention.html.

Contacto

Gibran Fuentes Pineda
Investigador Asociado "C", IIMAS, UNAM

gibranfp@unam.mx

LinkedIn: @gibranfuentespineda