



Who's Next?

D. Comanducci

Introduzione

RNN

Transformers

Self-attention

Transformer Layer

Positional Encoding

Il decoder

Conclusioni

Bibliografia

Who's Next? La predizione nell'era dell'attenzione

Dario Comanducci

Tesina per il Corso di Text Mining and Natural Language Processing

Master in Data Science and Statistical Learning
Università degli Studi di Firenze

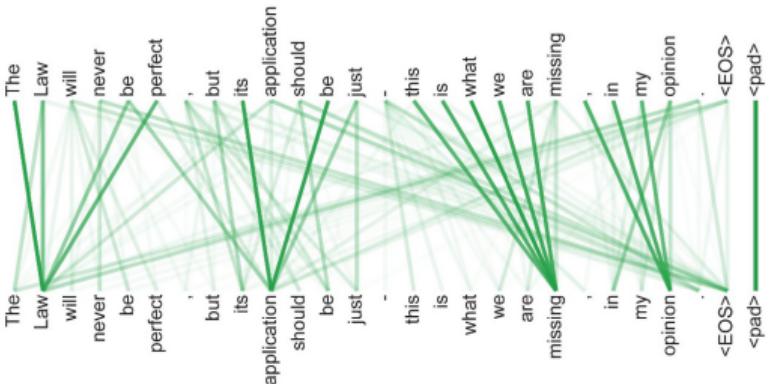
3 Dicembre 2024



L'attenzione

- ▶ Meccanismo alla base dei *transformers*
- ▶ Introdotta in NLP nell'ambito delle RNN per la traduzione automatica
- ▶ Estensione ad altri domini:
 - ▶ proteine (sequenze di ammino-acidi)
 - ▶ audio (segnali temporali campionati)
 - ▶ computer vision (classificazione, generazione di immagini)

"The Law will never be perfect, but its application should be just – this is what we are missing, in my opinion."



Who's Next?

D. Comanducci

Introduzione

RNN

Transformers

Self-attention

Transformer Layer

Positional Encoding

Il decoder

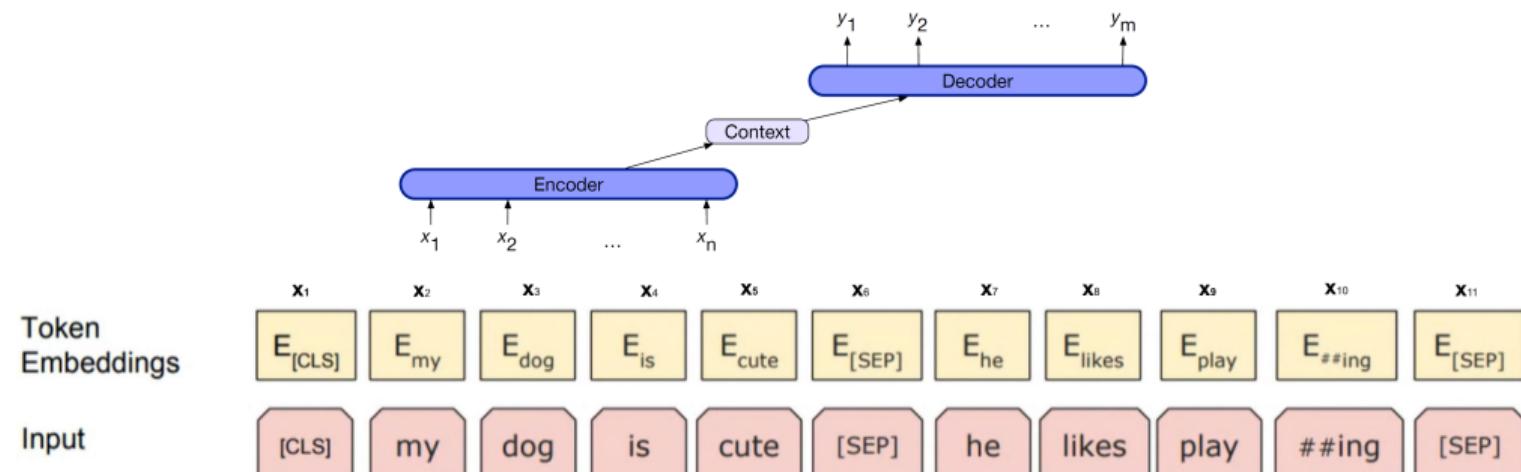
Conclusioni

Bibliografia



Encoder-Decoder

- ▶ la struttura Encoder-Decoder viene impiegata in task “sequence-to-sequence”: una sequenza in input viene convertita in un’altra (es. traduzione automatica)
- ▶ Un *encoder* trasforma gli embeddings in rappresentazioni contestualizzate (*contesto*) a supporto di una varietà di attività
- ▶ Un *decoder* sfrutta il contesto per generare l’uscita dell’attività specifica



Who's Next?

D. Comanducci

Introduzione

RNN

Transformers

Self-attention

Transformer Layer

Positional Encoding

Il decoder

Conclusioni

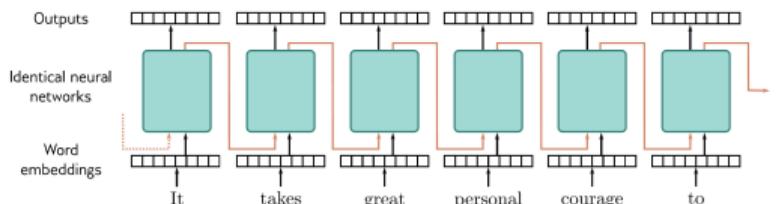
Bibliografia



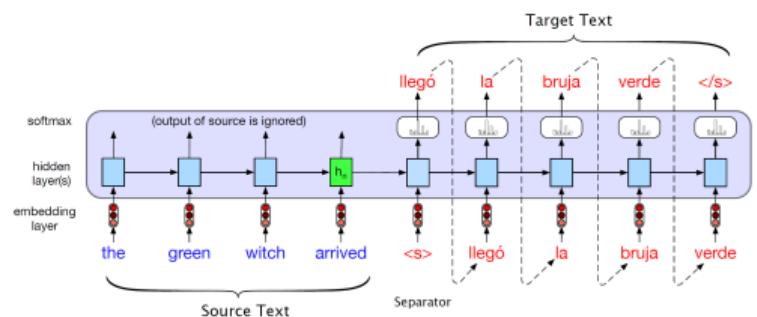
Attenzione con Reti RNN

Modello encoder-decoder senza attenzione

Modello RNN



Encoder-decoder con RNN



- ▶ ciascun blocco è una MLP
- ▶ concatenazione di N blocchi con stessi pesi condivisi in W, U, V
- ▶ lo *stato interno \mathbf{h}* di un blocco viene passato al successivo
- ▶ l'uscita \mathbf{y}_k del blocco k° dipende dall'ingresso corrente \mathbf{x}_k e da \mathbf{h}_{k-1}
- ▶ token \mathbf{x}_k elaborati *sequenzialmente*

$$\mathbf{h}_k = g(W\mathbf{x}_k + U\mathbf{h}_{k-1})$$

$$\mathbf{y}_k = f(V\mathbf{h}_k)$$

$$g(\cdot) = \text{ReLU}(\cdot), \tanh(\cdot) \dots$$

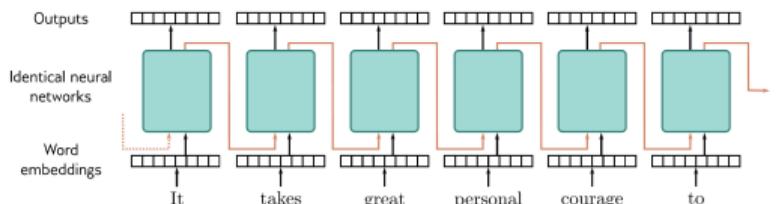
$$f(\cdot) = \text{softmax}(\cdot)$$



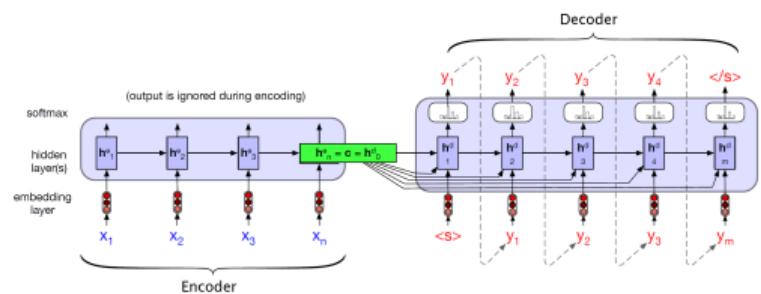
Attenzione con Reti RNN

Modello encoder-decoder senza attenzione

Modello RNN



Encoder-decoder con RNN



- ▶ ciascun blocco è una MLP
- ▶ concatenazione di N blocchi con stessi pesi condivisi in W, U, V
- ▶ lo *stato interno h* di un blocco viene passato al successivo
- ▶ l'uscita y_k del blocco k° dipende dall'ingresso corrente x_k e da h_{k-1}
- ▶ token x_k elaborati *sequenzialmente*

$$\mathbf{h}_k = g(\mathbf{W}\mathbf{x}_k + \mathbf{U}\mathbf{h}_{k-1})$$

$$\mathbf{y}_k = f(\mathbf{V}\mathbf{h}_k)$$

$$g(\cdot) = \text{ReLU}(\cdot), \tanh(\cdot) \dots$$

$$f(\cdot) = \text{softmax}(\cdot)$$

Who's Next?

D. Comanducci

Introduzione

RNN

Transformers

Self-attention

Transformer Layer

Positional Encoding

Il decoder

Conclusioni

Bibliografia

Attenzione con Reti RNN

Addestramento RNN



Self-supervised Learning + teacher forcing

In una sequenza $w_1 \dots w_t$ la parola w_j è nota date le precedenti $w_1 \dots w_{j-1}$

- ▶ $\mathbf{y}_j[w_k] = \hat{\mathbb{P}}[w_k = w_j | w_1 \dots w_{j-1}]$ **\mathbf{y}_j uscita del blocco j^o**
- ▶ $\mathbb{P}[w_k | w_1 \dots w_{j-1}] = \mathbf{1}(w_k = w_j)$ modella la vera distribuzione di $w_k \in V$
- ▶
$$L_{CE}(w_j) = \sum_{w_k \in V} \mathbb{P}[w_k = w_j | w_1 \dots w_{j-1}] \log \frac{1}{\hat{\mathbb{P}}[w_k = w_j | w_1 \dots w_{j-1}]} \\ = - \sum_{w_k \in V} \mathbf{1}(w_k = w_j) \log \hat{\mathbb{P}}[w_k | w_1 \dots w_{j-1}] = - \log \mathbf{y}_j[w_j]$$
- ▶ $\sum_j L_{CE}(w_j) = - \sum_j \log \mathbf{y}_j[w_j]$ **(cross-entropy loss da minimizzare)**

Interpretazione

- ▶ $\mathbb{P}[w_1 \dots w_t] = \mathbb{P}[w_1] \cdot \mathbb{P}[w_2 | w_1] \cdot \dots \cdot \mathbb{P}[w_t | w_1 \dots w_{t-1}]$
- ▶ $\hat{\mathbb{P}}[w_1 \dots w_t] = \prod_{i=1}^t \mathbf{y}_i[w_i]$ **verosimiglianza** della RNN come stimatore
- ▶ $RNN = \arg \min - \sum_j \log \mathbf{y}_j[w_j] = \arg \max \hat{\mathbb{P}}[w_1 \dots w_t]$ **max. verosimiglianza**

Who's Next?

D. Comanducci

Introduzione

RNN

Transformers

Self-attention

Transformer Layer

Positional Encoding

Il decoder

Conclusioni

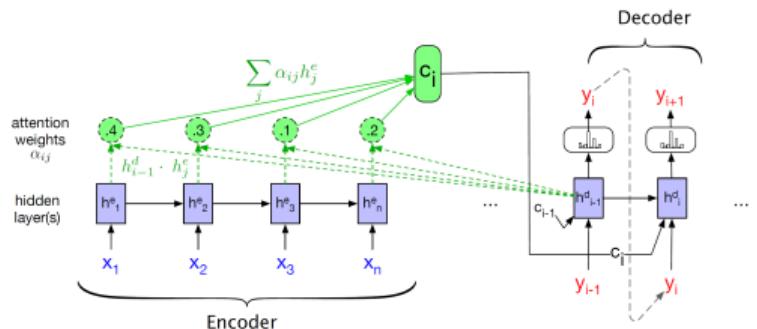
Bibliografia



Attenzione con Reti RNN

Modello encoder-decoder *con* attenzione

Encoder-decoder con RNN + attenzione



\mathbf{h}_j^e : stato interno (blocco # j) dell'encoder

\mathbf{h}_i^d : stato interno (blocco # i) del decoder

$$\mathbf{c}_i = \sum_j \underbrace{\frac{\exp(\mathbf{h}_{i-1}^{d\top} \mathbf{h}_j^e)}{\sum_k \exp(\mathbf{h}_{i-1}^{d\top} \mathbf{h}_k^e)}}_{\alpha_{ij}} \mathbf{h}_j^e$$

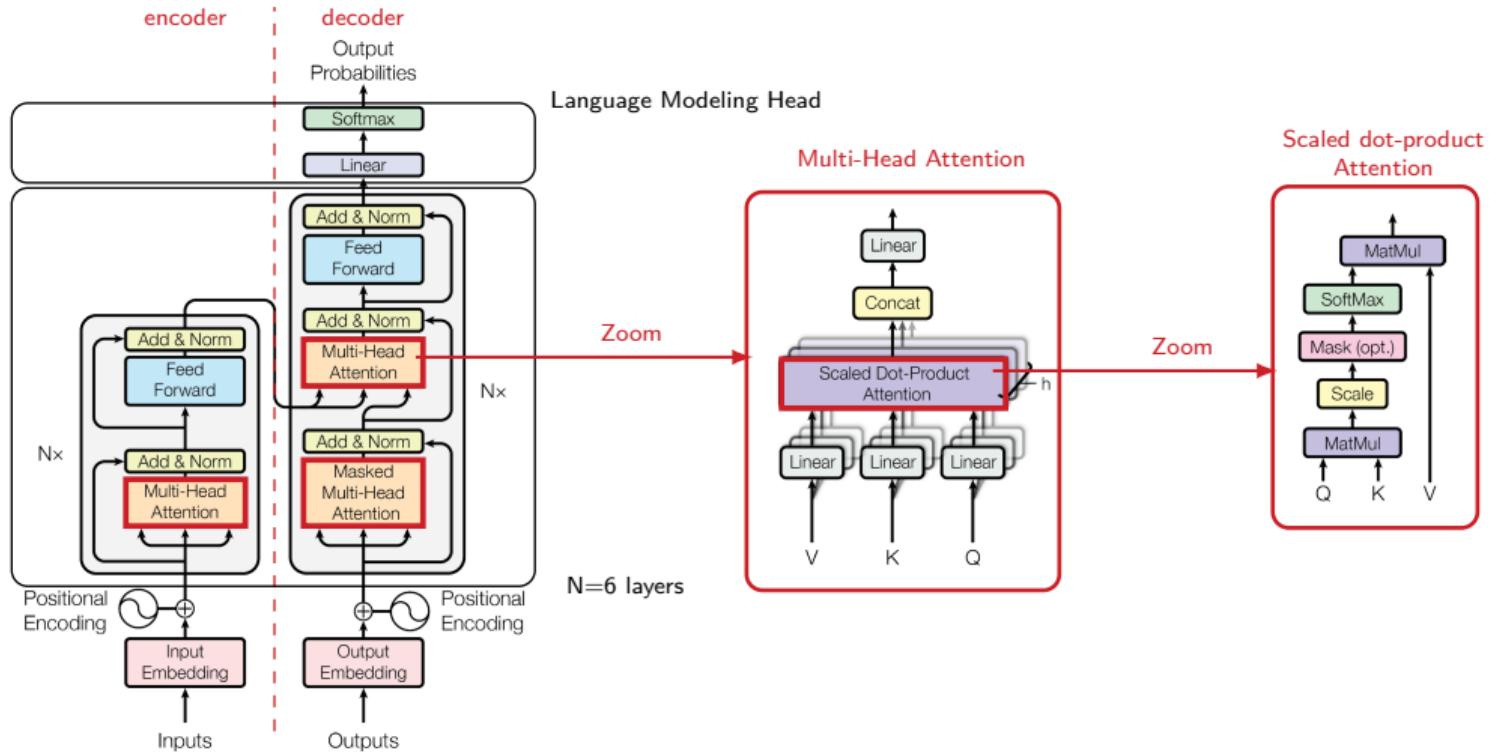
dot-product attention

Per ogni blocco i del decoder, $\mathbf{h}_{i-1}^{d\top} \mathbf{h}_j^e$ fornisce la **similarità** tra lo stato precedente \mathbf{h}_{i-1}^d del decoder rispetto agli stati interni dell'encoder, normalizzata poi in α_{ij} tramite softmax rispetto a tutti gli stati dell'encoder



Transformers

Architettura generale



Who's Next?

D. Comanducci

Introduzione

RNN

Transformers

Self-attention

Transformer Layer

Positional Encoding

Il decoder

Conclusioni

Bibliografia



Dot-product self-attention

- ▶ inputs: $\mathbf{x}_1 \dots \mathbf{x}_N \in \mathbb{R}^D$
- ▶ outputs: $\mathbf{sa}_1 \dots \mathbf{sa}_N \in \mathbb{R}^d$
- ▶ value $\mathbf{v}_n = \beta_v + \Omega_v \mathbf{x}_n \in \mathbb{R}^d$
- ▶ key $\mathbf{k}_n = \beta_k + \Omega_k \mathbf{x}_n \in \mathbb{R}^D$
- ▶ query $\mathbf{q}_n = \beta_q + \Omega_q \mathbf{x}_n \in \mathbb{R}^D$

self-attention: la stessa sequenza è usata per determinare le query, le chiavi e i valori

Information retrieval

Servizio di selezione film

- ▶ **value**: i film in catalogo
- ▶ **key**: attributi di ogni film (genere, cast...)
- ▶ **query**: preferenze dell'utente
- ▶ Il servizio di selezione confronta il vettore di *query* con la *key* di ogni film per trovare la corrispondenza migliore e segnalare il film all'utente sotto forma di *value*

Hard-attention: l'utente “presta attenzione” al film la cui chiave è più simile alla sua query

Who's Next?

D. Comanducci

Introduzione

RNN

Transformers

Self-attention

Transformer Layer

Positional Encoding

Il decoder

Conclusioni

Bibliografia



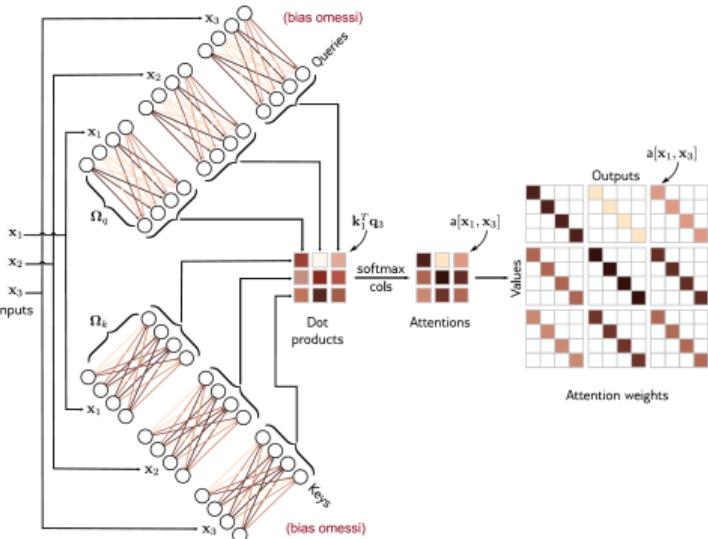
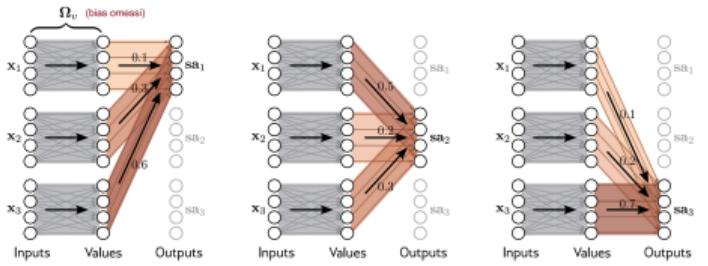
Dot-product self-attention

- ▶ inputs: $x_1 \dots x_N \in \mathbb{R}^D$
- ▶ outputs: $sa_1 \dots sa_N \in \mathbb{R}^d$
- ▶ value $v_n = \beta_v + \Omega_v x_n \in \mathbb{R}^d$
- ▶ key $k_n = \beta_k + \Omega_k x_n \in \mathbb{R}^D$
- ▶ query $q_n = \beta_q + \Omega_q x_n \in \mathbb{R}^D$

self-attention: la stessa sequenza è usata per determinare le query, le chiavi e i valori

$$sa_n(x_1 \dots x_N) = \sum_m \underbrace{\frac{\exp k_m^\top q_n}{\sum_i \exp k_i^\top q_n}}_{a[x_m, x_n]} v_m$$

(soft-attention)



Who's Next?

D. Comanducci

Introduzione

RNN

Transformers

Self-attention

Transformer Layer

Positional Encoding

Il decoder

Conclusioni

Bibliografia



Scaled dot-product attention

Notazione matriciale

$$\mathbf{X} = [\mathbf{x}_1 \dots \mathbf{x}_n] \in \mathbb{R}^{D \times N}$$

$$\mathbf{1} = [1 \dots 1]^\top \in \mathbb{R}^N$$

$$\mathbf{V} = \beta_v \mathbf{1}^\top + \Omega_v \mathbf{X}$$

$$\mathbf{K} = \beta_\kappa \mathbf{1}^\top + \Omega_\kappa \mathbf{X}$$

$$\mathbf{Q} = \beta_q \mathbf{1}^\top + \Omega_q \mathbf{X}$$

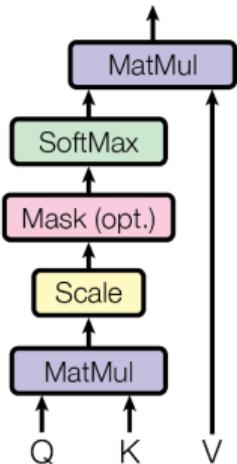
$$\text{Sa}(\mathbf{X}) = \mathbf{V} \cdot \text{softmax}(\mathbf{K}^\top \mathbf{Q})$$

Scaled dot-product self-attention

$$\mathbf{H} = \mathbf{V} \cdot \text{softmax} \left(\frac{\mathbf{K}^\top \mathbf{Q}}{\sqrt{D}} \right)$$

normalizziamo con la “dev. std. \sqrt{D} ”

Blocco Scaled Dot-Product Attention



- ▶ i gradienti della softmax tendono esponenzialmente a 0 per input grandi
- ▶ se query e chiavi fossero indip. casuali a media zero e varianza 1, la varianza del prodotto scalare sarebbe D



Scaled dot-product attention

Notazione matriciale

$$\mathbf{X} = [\mathbf{x}_1 \dots \mathbf{x}_n] \in \mathbb{R}^{D \times N}$$

$$\mathbf{1} = [1 \dots 1]^\top \in \mathbb{R}^N$$

$$\mathbf{V} = \beta_v \mathbf{1}^\top + \Omega_v \mathbf{X}$$

$$\mathbf{K} = \beta_k \mathbf{1}^\top + \Omega_k \mathbf{X}$$

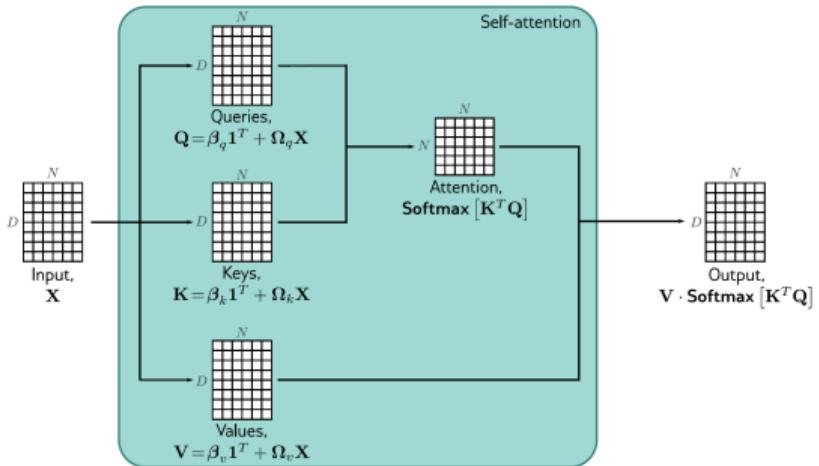
$$\mathbf{Q} = \beta_q \mathbf{1}^\top + \Omega_q \mathbf{X}$$

$$\text{Sa}(\mathbf{X}) = \mathbf{V} \cdot \text{softmax}(\mathbf{K}^\top \mathbf{Q})$$

Scaled dot-product self-attention

$$\mathbf{H} = \mathbf{V} \cdot \text{softmax} \left(\frac{\mathbf{K}^\top \mathbf{Q}}{\sqrt{D}} \right)$$

normalizziamo con la “dev. std. \sqrt{D} ”



- ▶ i gradienti della softmax tendono esponenzialmente a 0 per input grandi
- ▶ se query e chiavi fossero indip. casuali a media zero e varianza 1, la varianza del prodotto scalare sarebbe D

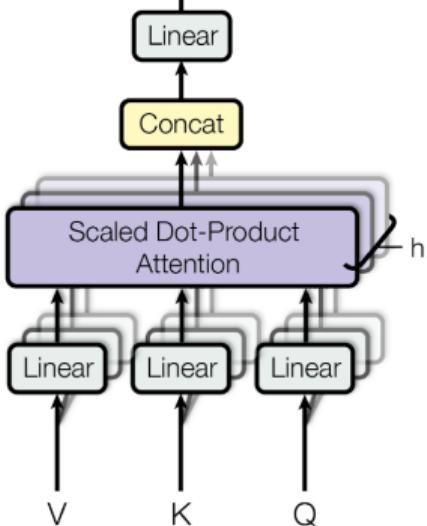


Multi-Head Attention

- ▶ Più modelli di attenzione rilevanti allo stesso tempo: per il tempo verbale, per il vocabolario...
- ▶ **Multi-Head Attention**: più blocchi Scaled Dot-Product Attention (*singole teste di attenzione*)
- ▶ Ogni testa ha parametri indipendenti per il calcolo di query, chiavi e valori
- ▶ H_h uscita dalla testa di attenzione h ($h = 1 \dots H$)

$$\mathbf{Y} = \Omega_c \begin{bmatrix} \mathbf{H}_1 \\ \vdots \\ \mathbf{H}_H \end{bmatrix} \quad (\mathbf{Y} \in \mathbb{R}^{D \times N})$$

Blocco Multi-Head Attention



- ▶ Per efficienza implementativa, con input $\mathbf{x}_n \in \mathbb{R}^D$ e H teste, di solito $\mathbf{v}_n, \mathbf{q}_n, \mathbf{k}_n \in \mathbb{R}^{D/H} \Rightarrow \mathbf{H}_h \in \mathbb{R}^{D/H \times N}$

Who's Next?

D. Comanducci

Introduzione

RNN

Transformers

Self-attention

Transformer Layer

Positional Encoding

Il decoder

Conclusioni

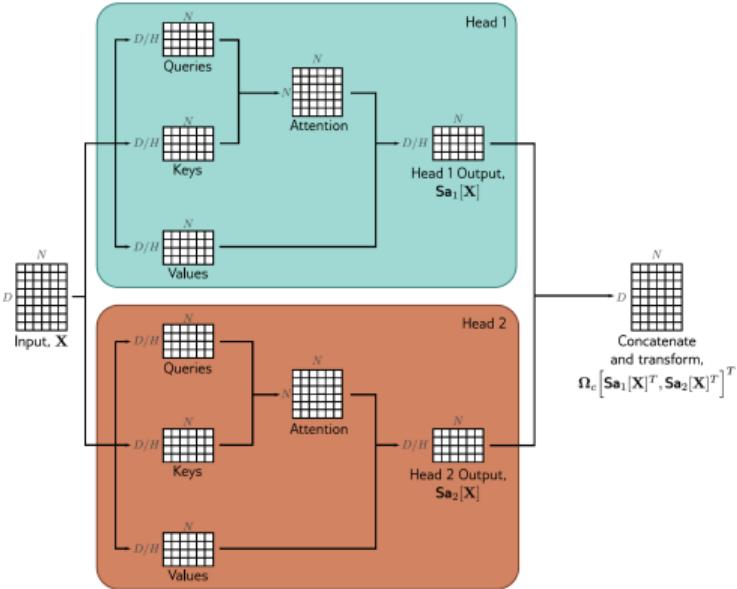
Bibliografia



Multi-Head Attention

- ▶ Più modelli di attenzione rilevanti allo stesso tempo: per il tempo verbale, per il vocabolario...
- ▶ **Multi-Head Attention:** più blocchi Scaled Dot-Product Attention (*singole teste di attenzione*)
- ▶ Ogni testa ha parametri indipendenti per il calcolo di query, chiavi e valori
- ▶ H_h uscita dalla testa di attenzione h ($h = 1 \dots H$)

$$\mathbf{Y} = \Omega_c \begin{bmatrix} \mathbf{H}_1 \\ \vdots \\ \mathbf{H}_H \end{bmatrix} \quad (\mathbf{Y} \in \mathbb{R}^{D \times N})$$



- ▶ Per efficienza implementativa, con input $\mathbf{x}_n \in \mathbb{R}^D$ e H teste, di solito $\mathbf{v}_n, \mathbf{q}_n, \mathbf{k}_n \in \mathbb{R}^{D/H} \Rightarrow \mathbf{H}_h \in \mathbb{R}^{D/H \times N}$

Who's Next?

D. Comanducci

Introduzione

RNN

Transformers

Self-attention

Transformer Layer

Positional Encoding

Il decoder

Conclusioni

Bibliografia



Transformer Layer

- ▶ 2 Residual connections
- ▶ 2 Layer di normalizzazione

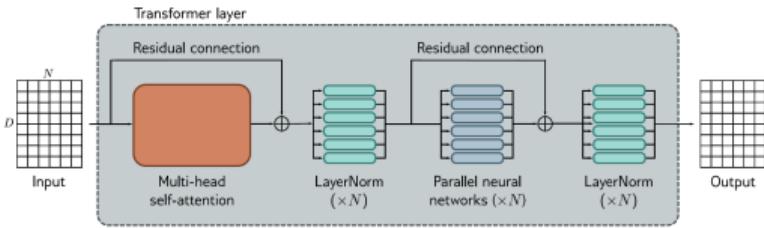
$$\text{Inorm}(\mathbf{u}_i) = \gamma \frac{\mathbf{u}_i - \mu_i}{\sigma_i} + \delta$$

$$\mu_i = \frac{1}{n} \sum_k u_{ik}$$

$$\sigma_i = \sqrt{\frac{1}{n} \sum_k (u_{ik} - \mu)^2}$$

(γ e δ appresi con l'addestramento)

- ▶ MLP: la stessa rete viene applicata separatamente a ciascuna delle N colonne



$$\mathbf{X} = [\mathbf{x}_1 \dots \mathbf{x}_N] \in \mathbb{R}^{D \times N}$$

$$\mathbf{Y} = \text{Multi-Head}(\mathbf{X}) \in \mathbb{R}^{D \times N}$$

$$\mathbf{Z}' = \mathbf{X} + \mathbf{Y} \quad (\mathbf{Z}' = [\mathbf{z}'_1 \dots \mathbf{z}'_N])$$

$$\mathbf{Z} = \text{Inorm}(\mathbf{Z}') \quad (\mathbf{Z} = [\mathbf{z}_1 \dots \mathbf{z}_N])$$

$$\mathbf{t}'_n = \mathbf{z}_n + \text{MLP}(\mathbf{z}_n) \quad (\mathbf{T}' = [\mathbf{t}'_1 \dots \mathbf{t}'_N])$$

$$\mathbf{T} = \text{Inorm}(\mathbf{T}') \quad (\mathbf{T} = [\mathbf{t}_1 \dots \mathbf{t}_N])$$

Who's Next?

D. Comanducci

Introduzione

RNN

Transformers

Self-attention

Transformer Layer

Positional Encoding

Il decoder

Conclusioni

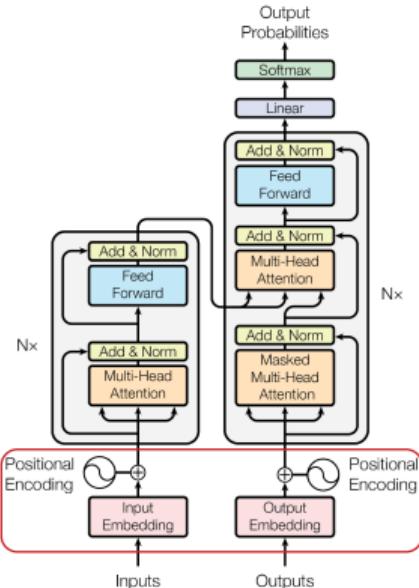
Bibliografia



Positional Embedding

In assenza di ricorsione, affinché un transformer possa utilizzare l'ordine della sequenza, occorre codificare le posizioni dei token in una rappresentazione tale che:

- ▶ sia univoca per ogni posizione
- ▶ sia limitata nei suoi valori
- ▶ generalizzi sulla lunghezza L delle sequenze
- ▶ indichi il numero di token tra due input



Who's Next?

D. Comanducci

Introduzione

RNN

Transformers

Self-attention

Transformer Layer

Positional Encoding

Il decoder

Conclusioni

Bibliografia



Positional Embedding

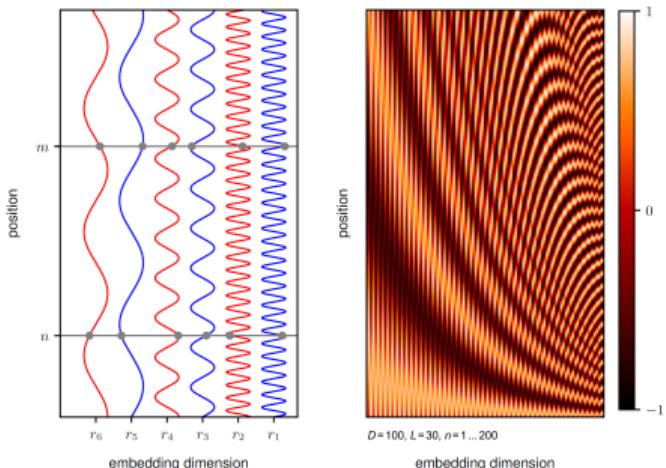
In assenza di ricorsione, affinché un transformer possa utilizzare l'ordine della sequenza, occorre codificare le posizioni dei token in una rappresentazione tale che:

- ▶ sia univoca per ogni posizione
- ▶ sia limitata nei suoi valori
- ▶ generalizzi sulla lunghezza L delle sequenze
- ▶ indichi il numero di token tra due input

$\tilde{\mathbf{x}}_n = \mathbf{x}_n + \mathbf{r}_n$, con $\mathbf{r}_n = [r_{n1} \dots r_{nD}]^\top \in \mathbb{R}^D$ la codifica del token \mathbf{x}_n in posizione n :

$$r_{ni} = \begin{cases} \sin \frac{n}{L^{i/D}} & i \text{ pari} \\ \cos \frac{n}{L^{(i-1)/D}} & i \text{ dispari} \end{cases}$$

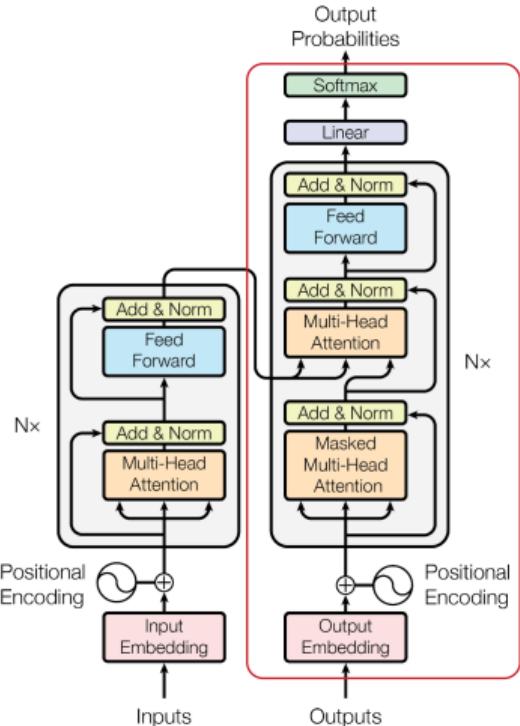
2 vettori casuali non correlati tendono ad essere ortogonali in spazi a grandi dimensioni
Le connessioni residue propagano \mathbf{r}_n attraverso i layer dei transformer





Lato decoder

- ▶ L'output finale dell'encoder è un set di vettori che rappresentano la sequenza di input con una ricca comprensione contestuale
- ▶ Questo output viene quindi utilizzato come contesto in input per il decoder
- ▶ Il decoder è autoregressivo, avviando il suo processo con un token di avvio
- ▶ La struttura del decoder è simile a quella dell'encoder, differenziandosi però in
 - ▶ masked self-attention
 - ▶ cross-attention
- ▶ Sopra la pila di layer è posto il Language Modeling Head, che mappa l'uscita in un vettore di probabilità sul vocabolario V



Who's Next?

D. Comanducci

Introduzione

RNN

Transformers

Self-attention

Transformer Layer

Positional Encoding

Il decoder

Conclusioni

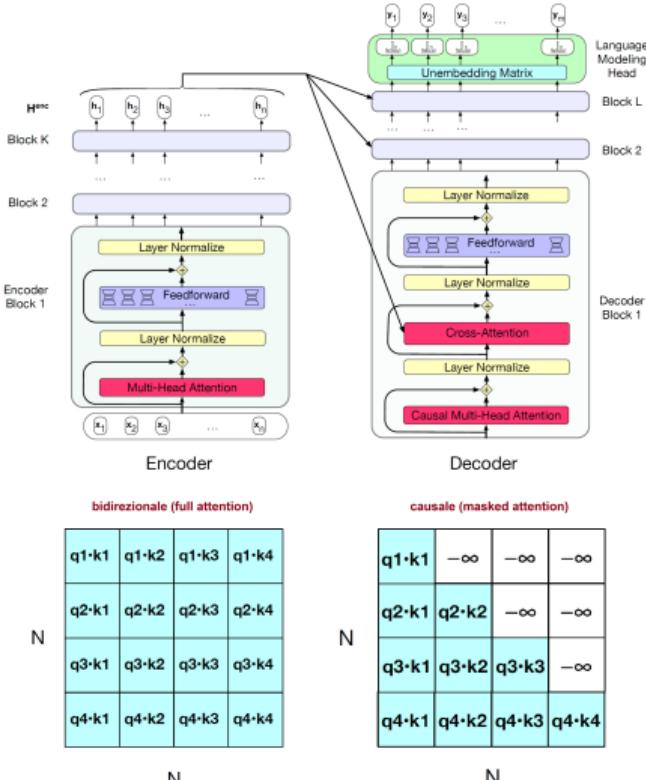
Bibliografia



Masked-attention

- ▶ Ognuno degli embedding di output rappresenta una frase parziale e per ognuna l'obiettivo è prevedere il token successivo nella sequenza
- ▶ Tramite la cross-entropy loss si massimizza la somma delle log-probabilità per il token successivo nella sequenza di ground-truth
- ▶ Ciascuna query non può guardare le chiavi future dei token

$$H = V_e \cdot \text{softmax} \left(\text{mask} \left(\frac{K^T Q}{\sqrt{D}} \right) \right)$$

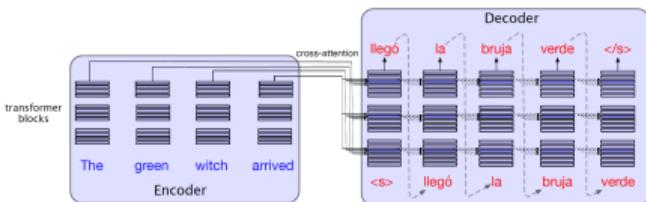
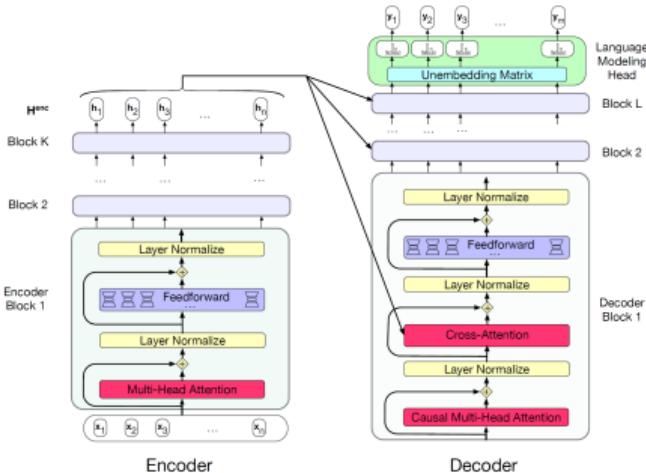
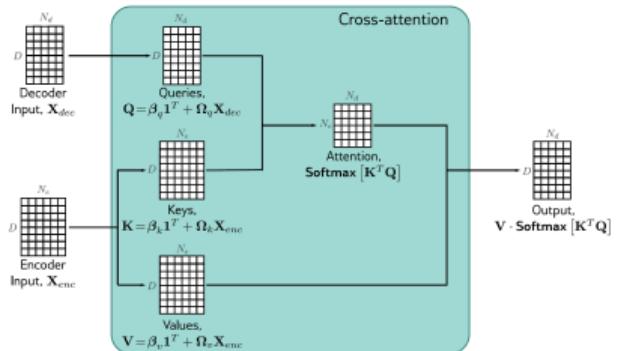


Cross-attention



- ▶ le query Q_d provengono dal livello precedente del decoder
 - ▶ le chiavi K_e e i valori V_e provengono dall'output dell'encoder

$$H = V_e \cdot \text{softmax} \left(\frac{K_e^\top Q_d}{\sqrt{D}} \right)$$





Conclusioni

Schema generale nel meccanismo di attenzione, con la terminologia dei transformer:

- ▶ uno dei dati in ingresso riveste il ruolo centrale di query \mathbf{q} , rispetto al quale enfatizzare o meno l'attenzione dagli altri dati (nel caso degli encoder-decoder basati su RNN la query è il corrente stato interno \mathbf{h} del decoder);
- ▶ il meccanismo di attenzione mappa una sequenza di N vettori $\{\mathbf{k}_n\}$, le chiavi, su una distribuzione di pesi $\mathbf{a} = [a_1 \dots a_N]$, dove i vettori \mathbf{k}_n codificano le caratteristiche dei dati su cui viene calcolata l'attenzione (nel caso degli encoder-decoder con RNN le chiavi sono i precedenti stati interni dell'encoder);
- ▶ la query \mathbf{q} per il dato i -esimo viene confrontata con ciascuna delle chiavi \mathbf{k}_j per definire uno score di similarità, su cui costruire una distribuzione di probabilità nei pesi a_{ij} del vettore di attenzione \mathbf{a}_i (softmax sugli score di similarità);
- ▶ rappresentazione biunivoca delle chiavi $\{\mathbf{k}_j\}$ anche in una forma di *value* $\{\mathbf{v}_j\}$ (nel caso dell'encoder-decoder con RNN questa diversificazione non esiste);
- ▶ i pesi in \mathbf{a}_i e $\{\mathbf{v}_j\}$ sono infine combinati in una codifica compatta del *contesto*.

Who's Next?

D. Comanducci

Introduzione

RNN

Transformers

Self-attention

Transformer Layer

Positional Encoding

Il decoder

Conclusioni

Bibliografia



Conclusioni

Schema generale nel meccanismo di attenzione, con la terminologia dei transformer:

- ▶ uno dei dati in ingresso riveste il ruolo centrale di query \mathbf{q} , rispetto al quale enfatizzare o meno l'attenzione dagli altri dati (nel caso degli encoder-decoder basati su RNN la query è il corrente stato interno \mathbf{h} del decoder);
- ▶ il meccanismo di attenzione mappa una sequenza di N vettori $\{\mathbf{k}_n\}$, le chiavi, su una distribuzione di pesi $\mathbf{a} = [a_1 \dots a_N]$, dove i vettori \mathbf{k}_n codificano le caratteristiche dei dati su cui viene calcolata l'attenzione (nel caso degli encoder-decoder con RNN le chiavi sono i precedenti stati interni dell'encoder);
- ▶ la query \mathbf{q} per il dato i -esimo viene confrontata con ciascuna delle chiavi \mathbf{k}_j per definire uno score di similarità, su cui costruire una distribuzione di probabilità nei pesi a_{ij} del vettore di attenzione \mathbf{a}_i (softmax sugli score di similarità);
- ▶ rappresentazione biunivoca delle chiavi $\{\mathbf{k}_j\}$ anche in una forma di *value* $\{\mathbf{v}_j\}$ (nel caso dell'encoder-decoder con RNN questa diversificazione non esiste);
- ▶ i pesi in \mathbf{a}_i e $\{\mathbf{v}_j\}$ sono infine combinati in una codifica compatta del *contesto*.

Who's Next?

D. Comanducci

Introduzione

RNN

Transformers

Self-attention

Transformer Layer

Positional Encoding

Il decoder

Conclusioni

Bibliografia



Conclusioni

Who's Next?

D. Comanducci

Introduzione

RNN

Transformers

Self-attention

Transformer Layer

Positional Encoding

Il decoder

Conclusioni

Bibliografia

Schema generale nel meccanismo di attenzione, con la terminologia dei transformer:

- ▶ uno dei dati in ingresso riveste il ruolo centrale di query \mathbf{q} , rispetto al quale enfatizzare o meno l'attenzione dagli altri dati (nel caso degli encoder-decoder basati su RNN la query è il corrente stato interno \mathbf{h} del decoder);
- ▶ il meccanismo di attenzione mappa una sequenza di N vettori $\{\mathbf{k}_n\}$, le chiavi, su una distribuzione di pesi $\mathbf{a} = [a_1 \dots a_N]$, dove i vettori \mathbf{k}_n codificano le caratteristiche dei dati su cui viene calcolata l'attenzione (nel caso degli encoder-decoder con RNN le chiavi sono i precedenti stati interni dell'encoder);
- ▶ la query \mathbf{q} per il dato i -esimo viene confrontata con ciascuna delle chiavi \mathbf{k}_j per definire uno score di similarità, su cui costruire una distribuzione di probabilità nei pesi a_{ij} del vettore di attenzione \mathbf{a}_i (softmax sugli score di similarità);
- ▶ rappresentazione biunivoca delle chiavi $\{\mathbf{k}_j\}$ anche in una forma di *value* $\{\mathbf{v}_j\}$ (nel caso dell'encoder-decoder con RNN questa diversificazione non esiste);
- ▶ i pesi in \mathbf{a}_i e $\{\mathbf{v}_j\}$ sono infine combinati in una codifica compatta del *contesto*.



Conclusioni

Who's Next?

D. Comanducci

Introduzione

RNN

Transformers

Self-attention

Transformer Layer

Positional Encoding

Il decoder

Conclusioni

Bibliografia

Schema generale nel meccanismo di attenzione, con la terminologia dei transformer:

- ▶ uno dei dati in ingresso riveste il ruolo centrale di query \mathbf{q} , rispetto al quale enfatizzare o meno l'attenzione dagli altri dati (nel caso degli encoder-decoder basati su RNN la query è il corrente stato interno \mathbf{h} del decoder);
- ▶ il meccanismo di attenzione mappa una sequenza di N vettori $\{\mathbf{k}_n\}$, le chiavi, su una distribuzione di pesi $\mathbf{a} = [a_1 \dots a_N]$, dove i vettori \mathbf{k}_n codificano le caratteristiche dei dati su cui viene calcolata l'attenzione (nel caso degli encoder-decoder con RNN le chiavi sono i precedenti stati interni dell'encoder);
- ▶ la query \mathbf{q} per il dato i -esimo viene confrontata con ciascuna delle chiavi \mathbf{k}_j per definire uno score di similarità, su cui costruire una distribuzione di probabilità nei pesi a_{ij} del vettore di attenzione \mathbf{a}_i (softmax sugli score di similarità);
- ▶ rappresentazione biunivoca delle chiavi $\{\mathbf{k}_j\}$ anche in una forma di *value* $\{\mathbf{v}_j\}$ (nel caso dell'encoder-decoder con RNN questa diversificazione non esiste);
- ▶ i pesi in \mathbf{a}_i e $\{\mathbf{v}_j\}$ sono infine combinati in una codifica compatta del *contesto*.



Conclusioni

Who's Next?

D. Comanducci

Introduzione

RNN

Transformers

Self-attention

Transformer Layer

Positional Encoding

Il decoder

Conclusioni

Bibliografia

Schema generale nel meccanismo di attenzione, con la terminologia dei transformer:

- ▶ uno dei dati in ingresso riveste il ruolo centrale di query \mathbf{q} , rispetto al quale enfatizzare o meno l'attenzione dagli altri dati (nel caso degli encoder-decoder basati su RNN la query è il corrente stato interno \mathbf{h} del decoder);
- ▶ il meccanismo di attenzione mappa una sequenza di N vettori $\{\mathbf{k}_n\}$, le chiavi, su una distribuzione di pesi $\mathbf{a} = [a_1 \dots a_N]$, dove i vettori \mathbf{k}_n codificano le caratteristiche dei dati su cui viene calcolata l'attenzione (nel caso degli encoder-decoder con RNN le chiavi sono i precedenti stati interni dell'encoder);
- ▶ la query \mathbf{q} per il dato i -esimo viene confrontata con ciascuna delle chiavi \mathbf{k}_j per definire uno score di similarità, su cui costruire una distribuzione di probabilità nei pesi a_{ij} del vettore di attenzione \mathbf{a}_i (softmax sugli score di similarità);
- ▶ rappresentazione biunivoca delle chiavi $\{\mathbf{k}_j\}$ anche in una forma di *value* $\{\mathbf{v}_j\}$ (nel caso dell'encoder-decoder con RNN questa diversificazione non esiste);
- ▶ i pesi in \mathbf{a}_i e $\{\mathbf{v}_j\}$ sono infine combinati in una codifica compatta del *contesto*.



Riferimenti bibliografici

- ▶ D. Jurafsky and J. H. Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition with Language Models*, Online manuscript August 2024.
<https://web.stanford.edu/~jurafsky/slp3/>
- ▶ C. M. Bishop and H. Bishop. *Deep Learning – Foundations and Concepts*. Springer, 2023. (<https://www.bishopbook.com/>)
- ▶ S. J. D. Prince. *Understanding Deep Learning*. The MIT Press, 2023.
<http://udlbook.com>
- ▶ A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser and I. Polosukhin. *Attention is all you need*. Neural Information Processing Systems, 2017.
- ▶ A. Galassi, M. Lippi and P. Torroni. *Attention in Natural Language Processing*. IEEE Transactions on Neural Networks and Learning Systems, 32(10):4291-4308, 2021.

Who's Next?

D. Comanducci

Introduzione

RNN

Transformers

Self-attention

Transformer Layer

Positional Encoding

IL decoder

Conclusioni

Bibliografia