RETI NEURALI RICORRENTI (RNN)

PROGETTATE PER LAVORARE CON DATI **SEQUENZIALI** o **TEMPORALI** COME TESTO, AUDIO, VIDEO. LA LORO CARATTERISTICA É LA CAPACITÀ DI **MEMORIZZARE** LE INFO PRECEDENTI PER UN PERIODO DI TEMPO E UTILIZZARLE PER L'ELABORAZIONE.

LE RETI TRADIZIONALI COME LE FEED FORWARD ASSUMONO CHE GLI INPUT SIANO **INDIPENDENTI** TRA DI LORO. IN MOLTI CASI GLI INPUT HANNO UNA DIPENDENZA TEMPORALE o CONTESTUALE.

↳ LE RNN INTRODUCONO IL **MECCANISMO DI RICORRENZA** CHE CONSENTE ALLA RETE DI **RICORDARE** IL CONTESTO **PRECEDENTE** DURANTE L'ELABORAZIONE DI UNA SEQUENZA.

LE RNN ELABORANO GLI ELEMENTI IN MANIERA SEQUENZIALE PASSO PER PASSO, AD OGNI PASSO TEMPORALE LA RETE CALCOLA LO **STATO NASCOSTO** h_t CHE CONTIENE INFORMAZIONI SULL'INPUT X_t E SULLO STATO NASCOSTO PRECEDENTE (h_{t-1}). QUESTO STATO NASCOSTO RAPPRESENTA UNA MEMORIA CHE SI AGGIORNA NEL TEMPO.

↳ NELLO STATO FINALE VERRÀ USATO PER GENERARE UN OUTPUT, L'OUTPUT PUÓ ESSERE GENERATO SIA PASSO x PASSO CHE SOLO ALLA FINE.

COSA CONTIENE h_{t-1} ? → INFORMAZIONI PASSATE CHE LA RETE HA "IMPARATO", PATTERN RILEVANTI / CORRELAZIONI.

ESEMPIO

FRASE = "IL GATTO DORME"

1. h_1: Contiene una rappresentazione della parola "Il".
2. h_2: Contiene una rappresentazione del contesto "Il gatto".
3. h_3: Contiene una rappresentazione del contesto "Il gatto dorme".

↳ UTILE x PREDIRE COSA POTREBBE VENIRE DOPO, AD ESEMPIO "SUL" SE LA FRASE FOSSE "IL GATTO DORME SUL DIVANO".

RETI NEURALI RICCORRENTI BIDIREZIONALI (BRNN)

RNN → L'OUTPUT AD UN DETERMINATO PASSO t DIPENDE SOLO DAGLI STATI PRECEDENTI. TUTTAVIA IN MOLTI PROBLEMI SEQUENZIALI È UTILE CONSIDERARE ANCHE IL CONTESTO FUTURO. AD ESEMPIO PER COMPRENDERE IL SIGNIFICATO DI UNA PAROLA È SPESSO NECESSARIO CONSIDERARE SIA LE PAROLE PRECEDENTI CHE SUCCESSIVE.

↳ LE RNN BIDIREZIONALI PROCESSANO LA SEQUENZA IN DUE DIREZIONI CONTEMPORANEAMENTE: AVANTI (FORWARD) e DIETRO (BACKWORD).
IN QUESTO MODO IN OGNI ISTANTE t PUÓ SFRUTTARE SIA INFORMAZIONI DAL PASSATO CHE DAL FUTURO.

↓

PER OGNI PASSO t SI OTTIENE:

- UNO STATO NASCOSTO $h_t^{FORWARD}$
- UNO STATO NASCOSTO $h_t^{BACKWARD}$

CHE VENGONO POI CONCATENATI PER FORMARE UNO STATO NASCOSTO BIDIREZIONALE.

⊖ DOPPIO DELLE RISORSE
⊖ NON FUNZIONANO IN TEMPO REALE POICHÉ NECESSITANO DELLA SEQUENZA INTERA

INTRODUZIONE AI TRANSFORMER

PROGETTATI PER RISOLVERE ALCUNE LIMITAZIONI DEI MODELLI SEQ2SEQ DI DEEP LEARNING BASATI SULL'ELABORAZIONE DEL TESTO IN SEQUENZA, CHE UTILIZZAVANO LE RETI RNN (RECURRENT NEURAL NETWORKS) O GLI LSTM.

LIMITAZIONI DEI MODELLI SEQ2SEQ

↳

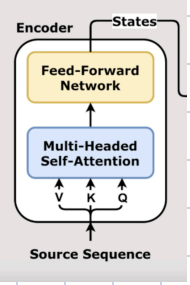
1. **Dipendenze a lungo termine difficili da apprendere**:
   - Gli RNN e LSTM elaborano i dati in modo sequenziale, il che significa che per calcolare l'output corrente devono processare passo dopo passo tutta la sequenza precedente.
   - Questo processo rende difficile catturare relazioni tra elementi distanti nella sequenza, portando alla perdita di informazioni quando la sequenza è molto lunga.
2. **Problema del gradiente vanishing/exploding**:
   - Nonostante gli LSTM abbiano mitigato in parte questo problema, i modelli sequenziali possono comunque perdere informazioni cruciali durante la propagazione del gradiente attraverso molteplici step temporali.
3. **Processamento lento**:
   - L'elaborazione sequenziale implica che le operazioni non possono essere parallelizzate, rallentando significativamente l'addestramento su grandi dataset.
4. **Bottleneck del contesto fisso**:
   - I modelli seq2seq spesso utilizzavano un "contesto fisso", cioè un unico vettore che riassumeva l'intera sequenza (nel caso del semplice encoder-decoder). Questo rendeva difficile catturare informazioni complesse e dettagliate.

COME FUNZIONANO I TRANSFORMER

SUPERANO QUESTE LIMITAZIONI UTILIZZANDO IL MECCANISMO DI ATTENZIONE, IN PARTICOLARE IL MULTI-HEAD ATTENTION.

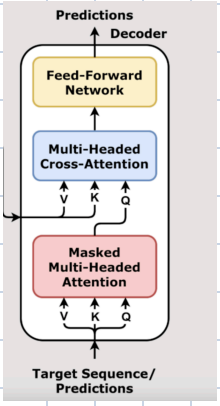
L'ARCHITETTURA È SUDDIVISA IN 2 COMPONENTI → ENCODER
→ DECODER

ENCODER → PRENDE IN INPUT UNA SEQUENZA E LA TRASFORMA IN UNA RAPPRESENTAZIONE ASTRATTA. FORMATO DA:



- MECCANISMO DI **SELF-ATTENTION** CHE È UTILIZZATO PER CALCOLARE LE RELAZIONI TRA TUTTE LE PAROLE NELLA SEQ. DI INPUT.
- RETE **FEED-FORWARD** COMPLETAMENTE CONNESSA, INTRODUCE NON LINEARITÀ.

DECODER → UTILIZZA QUESTA RAPPRESENTAZIONE PER GENERARE L'OUTPUT SEQUENZIALE (es. TRADUZ. DI UNA FRASE). FORMATO DA:



- **MASKED-MULTI-HEAD-ATTENTION**, SIMILE AL SELF-ATTENTION MA IMPEDISCE DI "VEDERE" LE COMPONENTI FUTURE, QUINDI OGNI PAROLA NON È INFLUENZATA DAI TOKEN FUTURI.
- **MULTI-HEADED-CROSS-ATTENTION**, IL DECODER USA LA CROSS ATTENTION PER CAPIRE QUALI PARTI DELL'INPUT SONO PIÙ IMPORTANTI PER GENERARE LA PROSSIMA PAROLA.
- RETE **FEED-FORWARD** COMPLETAMENTE CONNESSA, INTRODUCE NON LINEARITÀ.

MASKED LANGUAGE MODELING

TECNICA DI PRE-ADDESTRAMENTO X MODELLI DI LINGUAGGIO COME BERT. CONSISTE NEL "**MASCHERARE**" ALCUNE PAROLE IN UNA FRASE E ADDESTRARE IL MODELLO A **PREDIRE** LE PAROLE MANCANTI.
LE PAROLE DA MASCHERARE VENGONO SOSTITUITE CON UN TOKEN SPECIALE: [MASK]
ES.
"IL CANE [MASK] AL [MASK]" → "IL CANE **ABBAIA** AL **POSTINO**"

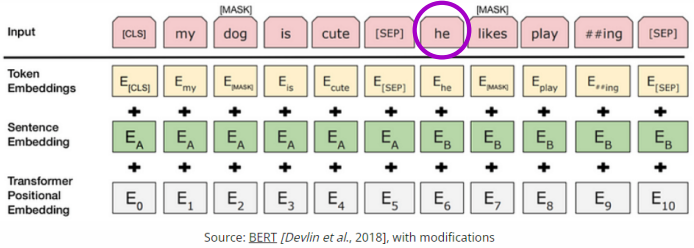
NEXT SENTENCE PREDICTION

TECNICA DI PRE-ADDESTRAMENTO USATA NEL MODELLO BERT, CONSISTE NELL'INSEGNARE AL MOD. A COMPRENDERE LA RELAZIONE TRA **DUE FRASI CONSECUTIVE**. VENGONO FORNITE AL MODELLO 2 FRASI E SI CHIEDE DI PREDIRE SE LA SECONDA FRASE **SEGUE** LA PRIMA NEL TESTO ORIGINALE. QUESTO COMPITO AIUTA IL MODELLO A CATTURARE LA **COERENZA DEL DISCORSO**, APPRENDENDO A DISTINGUERE TRA **COPPIE DI FRASI SEMANTICAMENTE CONNESSE** E **NON** CONNESSE.
IL MODELLO FORNIRÀ IN OUTPUT UNA TRA LE SEGUENTI CLASSI:

- IsNextSentence
- NotNextSentence

* IL "HE" È RIFERITO AL CANE DELLA 1° FRASE
SE AVESSI UN'ALTRA FRASE COME:
"STO ADDESTRANDO UNA RETE NEURALE" IL MODELLO

*



Source: BERT *[Devlin et al., 2018]*, with modifications

DEVE IMPARARE CHE NON C'ENTRA NULLA CON LA 1° FASE.