



# Masked Language Models

CORSO DI NATURAL LANGUAGE PROCESSING (ELABORAZIONE DEL LINGUAGGIO NATURALE)

a.a. 2025/2026

Prof. Roberto Pirrone

# Transformer encoder bidirezionali

# Transformer encoder bidirezionale

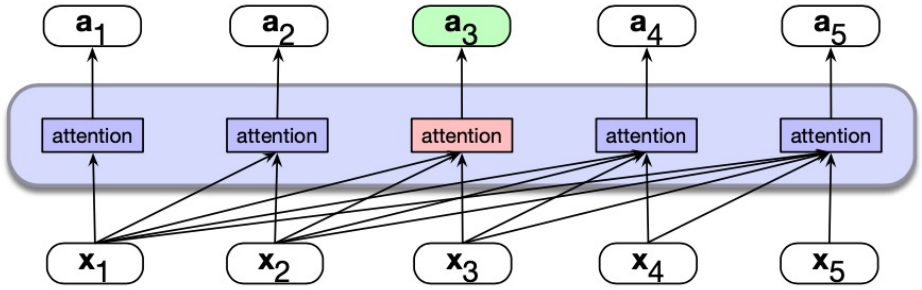
- Abbiamo imparato il *causal language modeling*
  - Esegue la predizione *autoregressiva* delle parole
  - *Transformers di tipo left-to-right*

- Ora introduciamo il transformer encoder bidirezionale (Bidirectional Transformer Encoder)
  - Vede sia il contesto sinistro che quello destro
  - Addestrato con il *masked language modeling*

# Transformer encoder bidirezionale

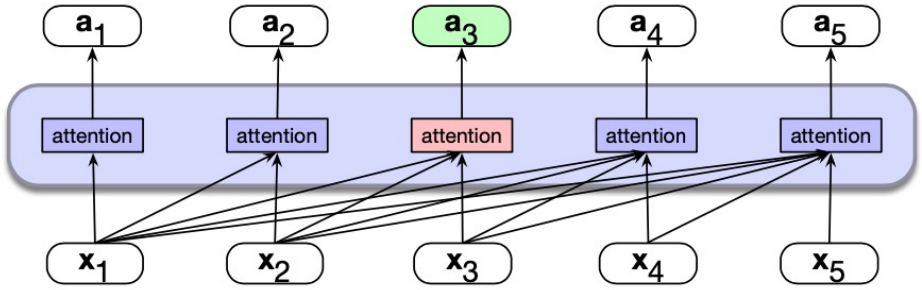
- BERT, RoBERTa, SpanBERT, XLM …
  - Sono utilizzati in applicazioni «a valle» (downstream) attraverso il *fine-tuning*
    - LM preaddestrati sono usati come ingresso per una rete posta sull'ultimo layer che viene addestrata per il task
    - Il processo di pretraining/fine-tuning è un tipo di *transfer learning*
  - Gli embedding generati dai Masked Language Model (MLM) sono *contestuali*

# Transformer encoder bidirezionale

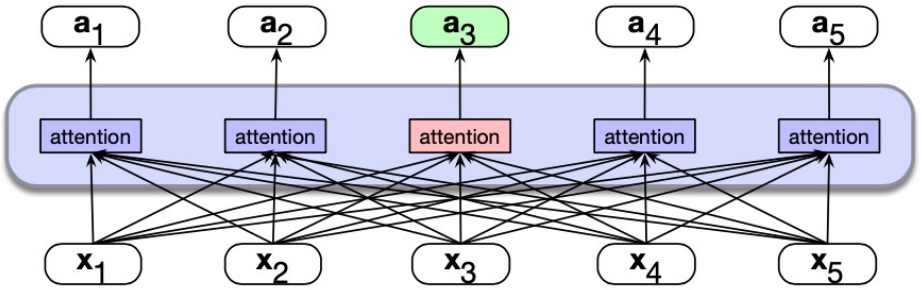



Architettura transformer left-to-right

# Transformer encoder bidirezionale




Left-to-right transformer architecture




Arichitettura transformer encoder bidirezionale

# Transformer encoder bidirezionale

- Meccanismo classico di multi-head self-attention

Single Head Attention

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

$$\mathbf{y}_i = \sum_{j=1}^{n} \alpha_{ij}\mathbf{v}_j \qquad \alpha_{ij} = \frac{\exp(\text{score}_{ij})}{\sum_{k=1}^{n}\exp(\text{score}_{ik})}$$

$$\text{score}_{ij} = \mathbf{q}_i \cdot \mathbf{k}_j$$

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

Multi-head Attention

Università degli Studi di Palermo

dipartimento di ingegneria unipa

CHILAB

# Transformer encoder bidirezionale

- Implementazione parallela della self-attention

N

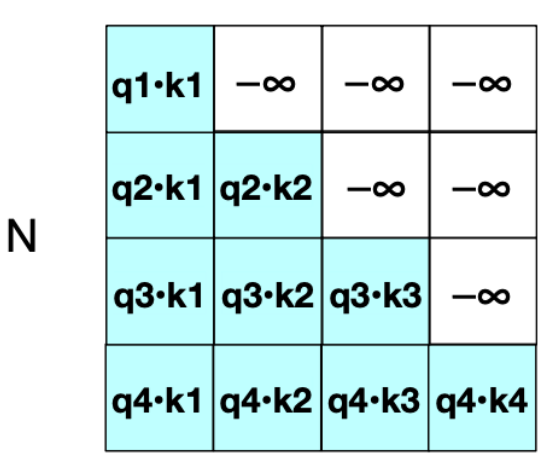
| | | | |
|---|---|---|---|
| q1·k1 | −∞ | −∞ | −∞ |
| q2·k1 | q2·k2 | −∞ | −∞ |
| q3·k1 | q3·k2 | q3·k3 | −∞ |
| q4·k1 | q4·k2 | q4·k3 | q4·k4 |

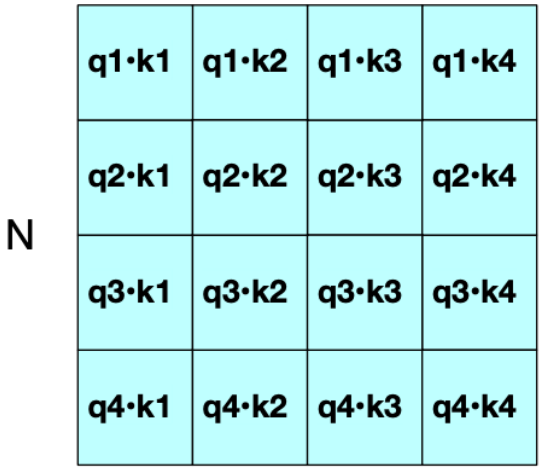
N

$$\mathbf{A} = \text{softmax}\left(\text{mask}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}}\right)\right)\mathbf{V}$$

# Transformer encoder bidirezionale

- Implementazione parallela della self-attention




$$\mathbf{A} = \text{softmax}\left(\text{mask}\left(\frac{\mathbf{QK}^\top}{\sqrt{d_k}}\right)\right)\mathbf{V} \qquad \mathbf{A} = \text{softmax}\left(\frac{\mathbf{QK}^\top}{\sqrt{d_k}}\right)\mathbf{V}$$

Università degli Studi di Palermo

dipartimento di ingegneria unipa

CHILAB

# Bidirectional Encoder Representations from Transformers (BERT)

- Tokenizzazione subword
  - Algoritmo WordPiece (variante del BPE)
  - Vocabolario da 30.000 subword
  - Input fisso di 512 embedding (token + posizionali) per controllare la complessità computazionale
  - I requisiti di memoria crescono quadraticamente con la dimensione dell'input

# Bidirectional Encoder Representations from Transformers (BERT)

- Algoritmo WordPiece
  - Ad ogni passo sceglie il merge che accresce maggiormente la probabilità di un LM rispetto alla tokenizzaizone
  - Usa un carattere speciale all'inizio di ogni token

| | |
|---|---|
| **words:** | Jet makers feud over seat width with big orders at stake |
| **wordpieces:** | _J et _makers _fe ud _over _seat _width _with _big _orders _at _stake |

# Bidirectional Encoder Representations from Transformers (BERT)

- Algoritmo WordPiece

1. Inizializza il lessico wordpiece con tutti i caratteri del corpus di addestramento

2. Ripeti finché non resta esattamente il numero desiderato *V* di token:

   a) Addestra un LM a n-grammi sul corpus di addestramento usando l'insieme attuale dei wordpiece

   b) Considera l'insieme di tutti i possibili wordpiece che si ottengono concatenando due wordpiece del lessico corrente. Scgegli quello che massimizza la probabilità del LM relativa al corpus di addestramento.

# Bidirectional Encoder Representations from Transformers (BERT)

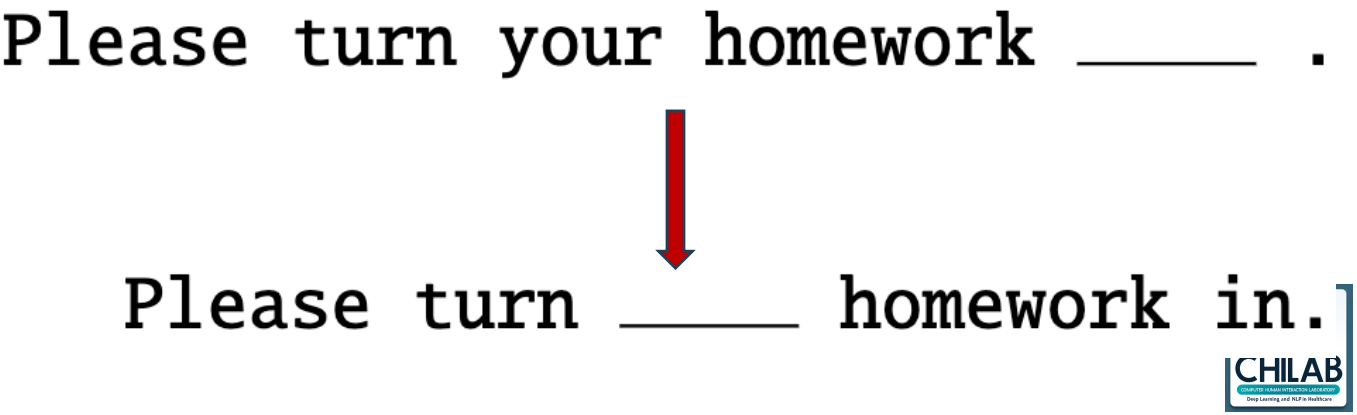
- Blocchi transformer multi-layer
  - 768 unità in ogni strato nascosto
  - $BERT_{BASE}$: 12 layer transformer, 12 teste di self-attention per layer, 110M parametri
  - $BERT_{LARGE}$: 24 layer transformer, 16 teste di self-attention per layer, 304M parametri

# Addestramento degli encoder bidirezionali

# Masked Language Modeling

- Il language modeling causale diventa banale se rimuoviamo la maschera causale

- Masked Language Modeling usa un approccio di «riempimento degli spazi vuoti»
  - The *cloze task*



# Masked Language Modeling

- Durante l'addestramento, al modello vengono rimossi uno o più elementi da una sequenza di input.
- Il modello deve quindi generare una distribuzione di probabilità sul vocabolario per ciascuno degli elementi mancanti.
- Successivamente, si utilizza la cross-entropy loss di ciascuna delle predizioni del modello per guidare il processo di apprendimento.

# Masked Language Modeling

- Il masking può essere generalizzato a *qualsiasi metodo che corrompe l'input di addestramento* e poi richiede al modello *di ricostruire l'input originale*.
    - Mascheramenti
    - Sostituzioni
    - Riordinamenti
    - Cancellazioni
    - Inserimenti di elementi estranei nel testo di addestramento

# Mascherare le parole

- BERT utilizza corpora testuali non annotati

- Un campione casuale di token viene selezionato per essere utilizzato nel task di addestramento

# Mascherare le parole

- Una volta selezionato, un token viene utilizzato in uno dei tre modi seguenti:
  - Viene sostituito con il token unico `[MASK]` del vocabolario
  - Viene sostituito con un altro token del vocabolario, scelto casualmente in base alle probabilità di unigramma dei token
  - Viene lasciato invariato

# Mascherare le parole

- In BERT il 15% dei tokenviene campionato per l'addestramento
    - 80% è sostituito dal token `[MASK]`
    - 10% è sostituito da un token casuale
    - 10% viene lasciato invariato

- Tutti i token partecipano al meccanismo di self-attention, ma solo quelli selezionati vengono utilizzati per calcolare la loss tramite la softmax.

# Mascherare le parole

$$L_{MLM} = -\frac{1}{|M|}\sum_{i \in M} \log P(x_i|\mathbf{h}_i^L)$$

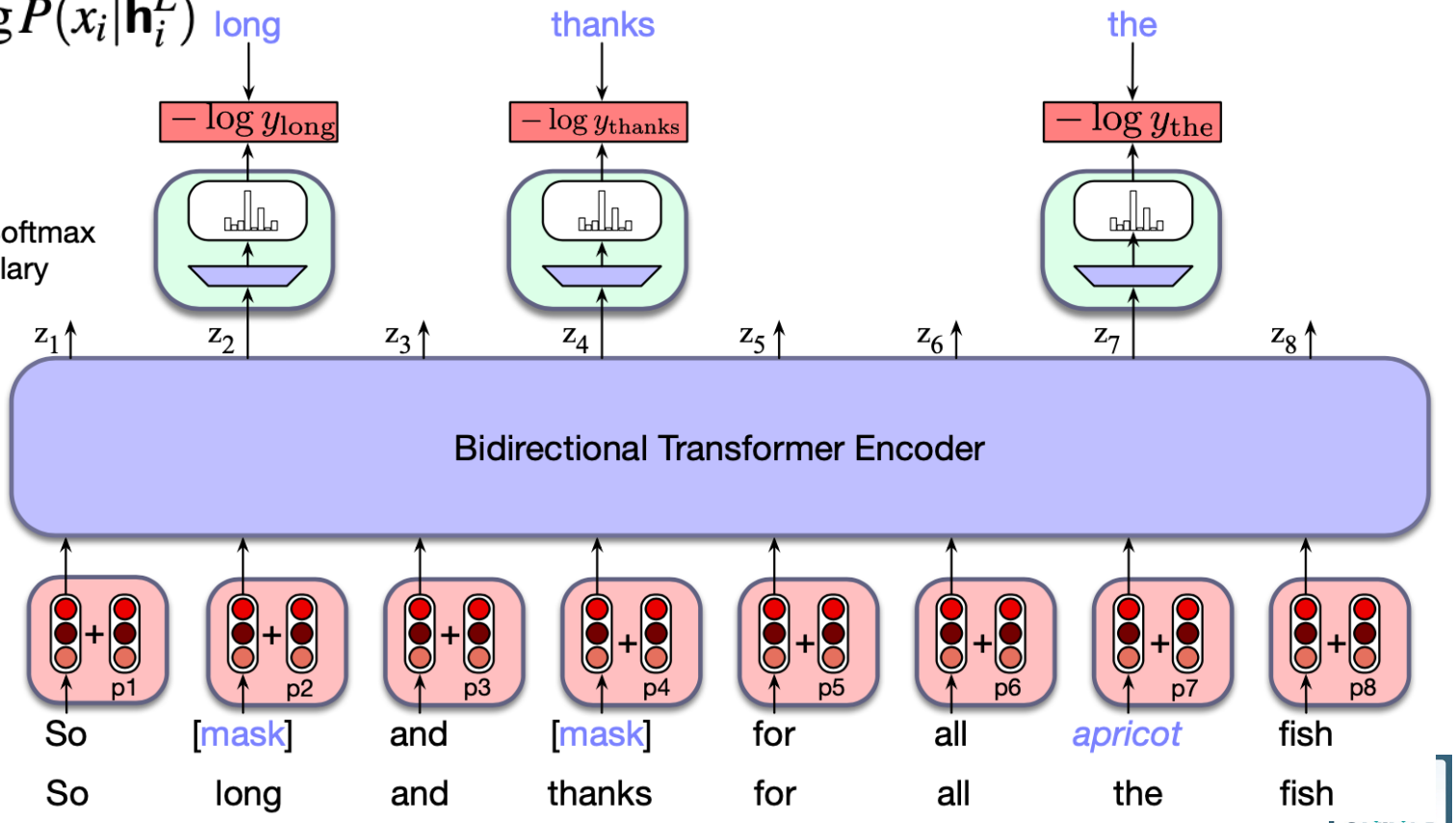
$$L_{MLM}(x_i) = -\log P(x_i|\mathbf{h}_i^L)$$

CE Loss

LM Head with Softmax over Vocabulary

$$\mathbf{u}_i = \mathbf{h}_i^L \mathbf{E}^T$$

$$\mathbf{y}_i = \text{softmax}(\mathbf{u}_i)$$




Token + Positional Embeddings

Università degli Studi di Palermo

CHILAB

# Mascherare gli span

- Molti compiti di NLP sono orientati alle frasi piuttosto che alle singole parole.

- In questo senso, il Masked Language Modeling orientato agli span (*Span-oriented MLM*) rappresenta una scelta migliore.

    - *Span*: una sequenza continua di una o più parole selezionate da un testo di addestramento, *prima della tokenizzazione in sotto-parole*.

# Mascherare gli span

- SpanBERT
  - Stessa percentuale di BERT per i token selezionati per l'addestramento (15%)
  - Si sceglie la lunghezza da una distribuzione geometrica polarizzata verso span brevi con un limite superiore di 10 parole

# Mascherare gli span

- SpanBERT
  - Data la lunghezza dello span, si campiona una posizione iniziale coerentemente con la lunghezza dell'input
  - Gli elementi nello span sono sostituiti usando lo stesso schema 80%, 10%, 10% di BERT

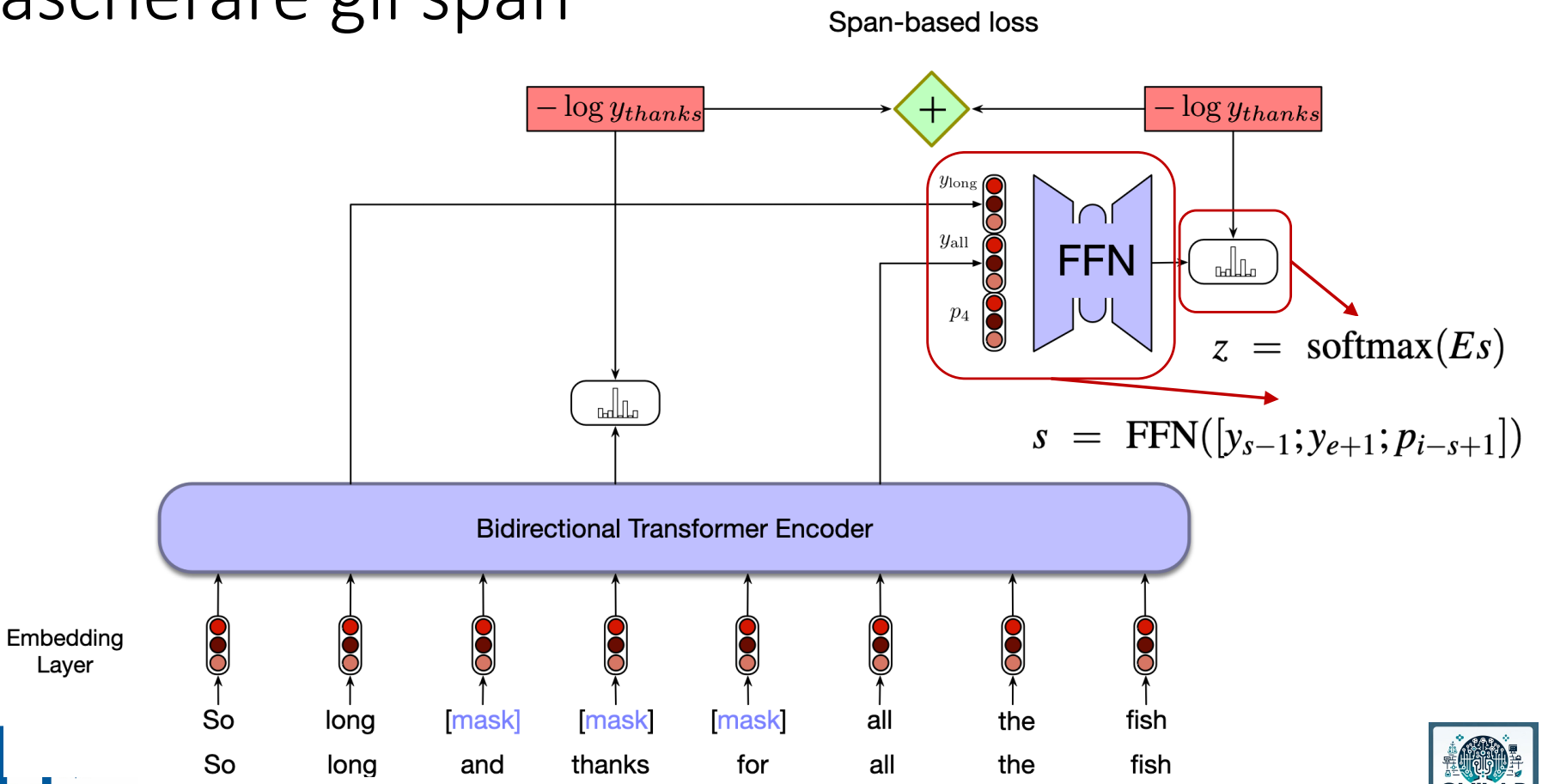
# Mascherare gli span

- La funzione di loss viene modificata per tener conto anche della cosiddetta *Span Boundary Objective* (SBO)

$$L(x) = L_{MLM}(x) + L_{SBO}(x)$$
$$L_{SBO}(x) = -\log P\left(x \mid x_{s-1}, x_{e+1}, p_x\right)$$

- $L_{SBO}$ viene appresa da una FFN a due layer

# Mascherare gli span



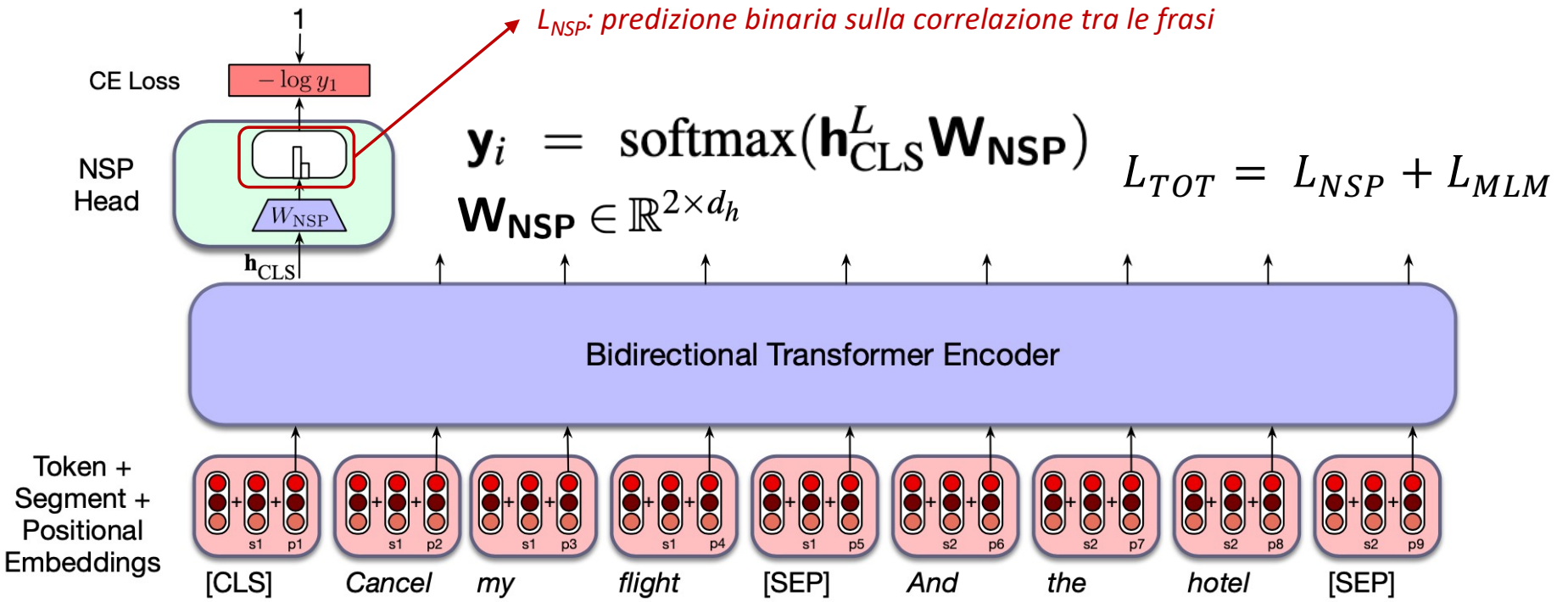
# Next sentence prediction

- Molti task di NLP includono la determinazione della relazione tra coppie di frasi
    - Individuazione di parafrasi (paraphrase detection)
    - Textual entailment (stabilire se il significato di una frase implica o contraddice quello dell'altra)
    - Coerenza del discorso tra due frasi adiacenti (discourse coherence)
    - …

# Next sentence prediction

- BERT esegue la predizione della frase successiva (Next Sentence Prediction - NSP) usando le seguenti procedure di addestramento:
  - Split 50% - 50% tra coppie positive di frasi (correlate) e negative (non correlate)
  - Vengono introdotti due nuovi token:
    - `[CLS]` inserito all'inizio della coppia di frasi
    - `[SEP]` inserito nel mezzo tra le due frasi e dopo il token finale

# Next sentence prediction



# Addestramento di BERT

- Addestrato su
    - BookCorpus (800M parole)
    - Wikipedia inglese (2500M parole)

- Addestramento della NSP usando lo schema 50% - 50%
    - Le coppie di frasi sono state campionate per entrare nella dimensione di input di 512 token

- 40 epoche di addestramento usando $L_{TOT} = L_{MLM} + L_{NSP}$

# Embedding contestuali (contextual embeddings)

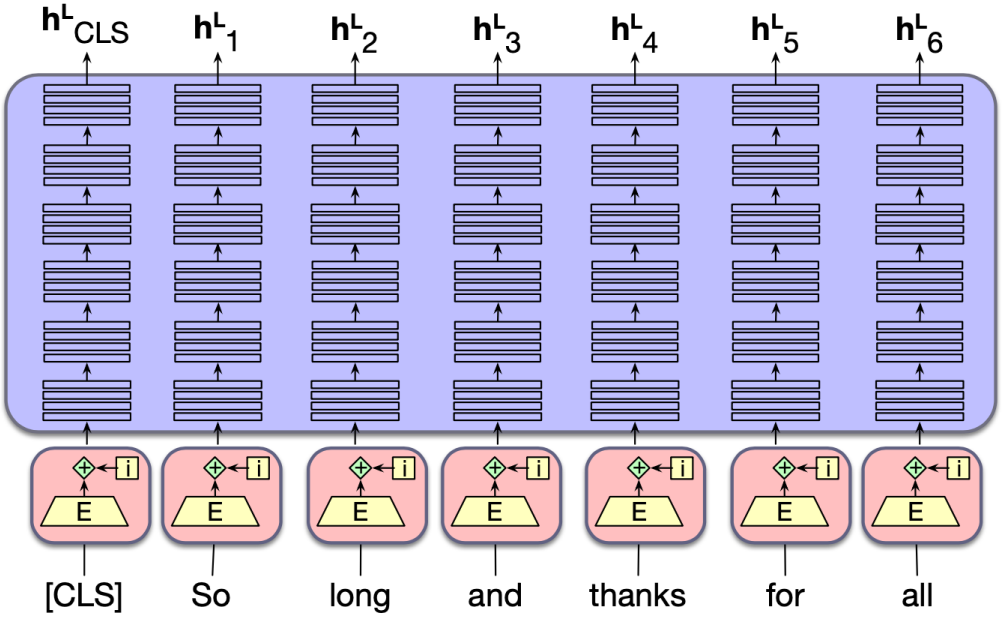
# Embedding contestuali

- L'uscita fornita dal layer finale di un LM pre-addestrato per ogni token $\boldsymbol{x}_i$ di una nuova frase può essere considerato un *embedding contestuale*

  - ***Embedding contestuali***: vettori che rappresentano alcuni aspetti del significato di un'*istanza di una parola* nel proprio contesto

  - Spesso il contextual embedding si ottiene facendo la media, token per token, dei corrispondenti stati nascosti negli ultimi quattro layer del modello

# Embedding contestuali




- Embedding statici → significato dei *tipi* delle parole
- Embedding contestuali → significato delle *istanze* delle parole

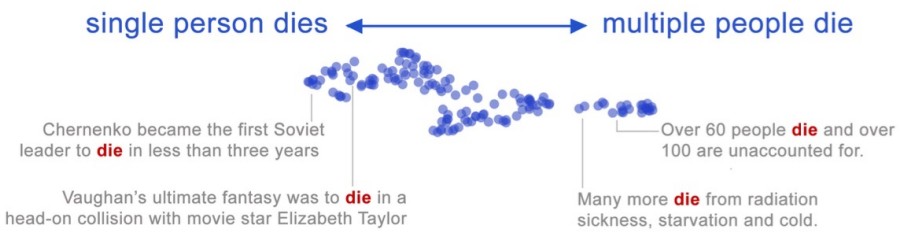
ContextualEmbedding(«long») → $\mathbf{h}^L_2$

# Embedding contestuali e senso delle parole

- Ricordiamo che le parole sono *ambigue*
  - Polisemia → molti significati per la stessa forma superficiale
  - I synset di WOrdNet

# Embedding contestuali e senso delle parole

- Ricordiamo che le parole sono *ambigue*



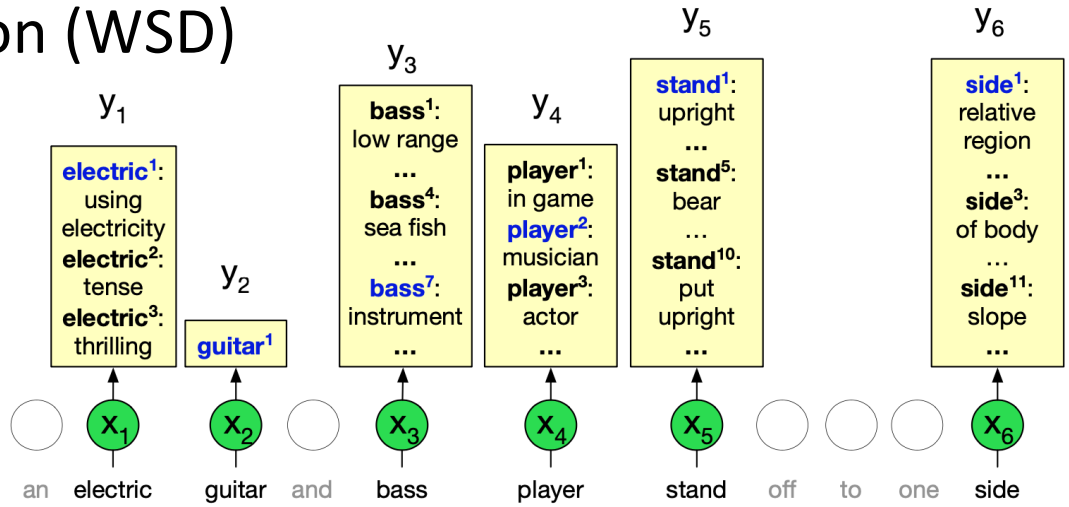
Università degli Studi di Palermo

dipartimento di ingegneria unipa

CHILAB

# Embedding contestuali e senso delle parole

- Word sense disambiguation (WSD)
  - Il task di scelta del senso corretto di una parola




  - Un algoritmo di WSD prende in ingresso la parola e i suoi synset di WordNet e restituisce il corretto senso della parola nel contesto.

# Embedding contestuali e senso delle parole

- 1-nearest neighbour WSD

1. Per ogni token *i* in un data set etichettato per i sensi (ad es. SenseEval)
   a) Calcola l'embedding contestuale $\mathbf{v}_i$

2. Per ogni senso *s* composto da *n* token
   a) Calcola l'*embedding di senso* $\mathbf{v}_s$

$$\mathbf{v}_s = \frac{1}{n}\sum_i \mathbf{v}_i \qquad \forall \mathbf{v}_i \in \text{tokens}(s)$$

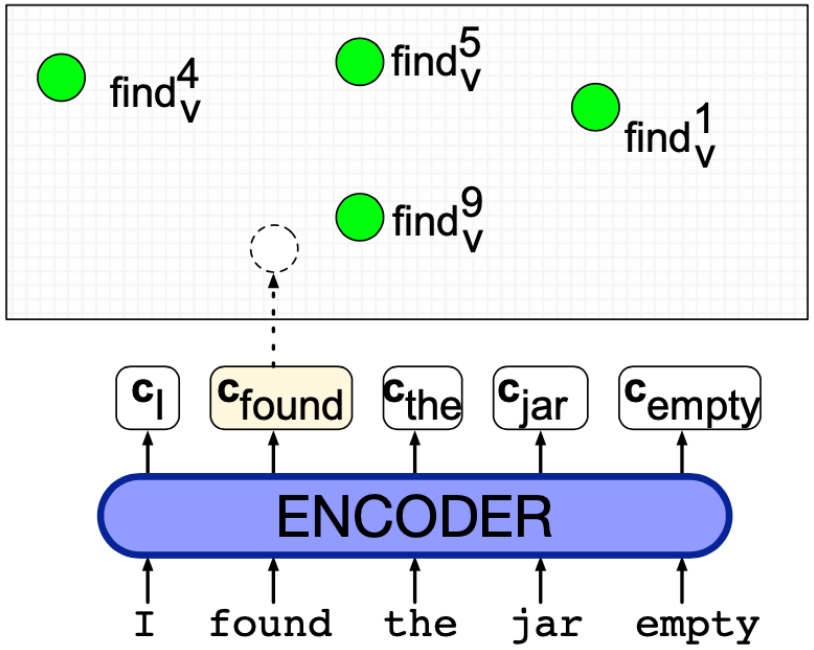
# Embedding contestuali e senso delle parole

- 1-nearest neighbour WSD

1. Data una parola target *t* in un contesto, calcola l'embedding contestuale **t**

2. Fornisci il senso come:

$$\text{sense}(t) = \underset{s \in \text{senses}(t)}{\text{argmax}} \ \text{cosine}(\mathbf{t}, \mathbf{v}_s)$$



# Embedding contestuali e Word Similarity

- Gli embedding contestuali forniti sia dai MLM sia dai LM autoregressivi sono *anisotropi*
    - Tendono più o meno tutti nella stessa direzione così che la loro distanza coseno e sempre circa pari a 1
    - In un embedding contestuale poche dimensioni dominano le altre

# Embedding contestuali e Word Similarity

- Gli embedding contestuali forniti sia dai MLM sia dai LM autoregressivi sono *anisotropi*
    - Si può tentare di standardizzarli con lo z-scoring, ma la distanza coseno tende a sottostimare il giudizio umano sulla WSD per parole molto frequenti
    - In generale sono utilizzati come ingressi per classificatori fine-tuned nelle applicazioni downstream

# Fine-tuning per i transformer encoder bidirezionali

# Fine-tuning di LM pre-addestrati

- I LM pre-addestrati sono bravi a generalizzare
  - Sono utili per le applicazioni downstream

- Diverse strategie di fine-tuning

  - *Prompt* fornito al modello per metterlo in una condizione in cui generi contestualmente ciò che è il nostro obiettivo (tipico dei LM decoder-only)

# Fine-tuning di LM pre-addestrati

- I LM pre-addestrati sono bravi a generalizzare
  - Sono utili per le applicazioni downstream

- Diverse strategie di fine-tuning

  - *Addestriamo nuovamente gli ultimi layer del modello* tipicamente con tecniche di Parameter Efficient Fine-Tuning (PEFT) che vedremo in seguito

# Fine-tuning di LM pre-addestrati

- I LM pre-addestrati sono bravi a generalizzare
  - Sono utili per le applicazioni downstream

- Diverse strategie di fine-tuning

  - *Aggiungiamo un circuito neurale* (una *testa*) in uscita del LM pre-addestrato che fornisce a quest'ultima le sue uscite

# Fine-tuning

- Quando si utilizza una testa specifica per l'applicazione
  - I nuovi parametri sono addestrati utilizzando dati *etichettati* propri dell'applicazione
  - I parametri del modello sono congelati o si consentono alcuni cambiamenti negli ultimi layer

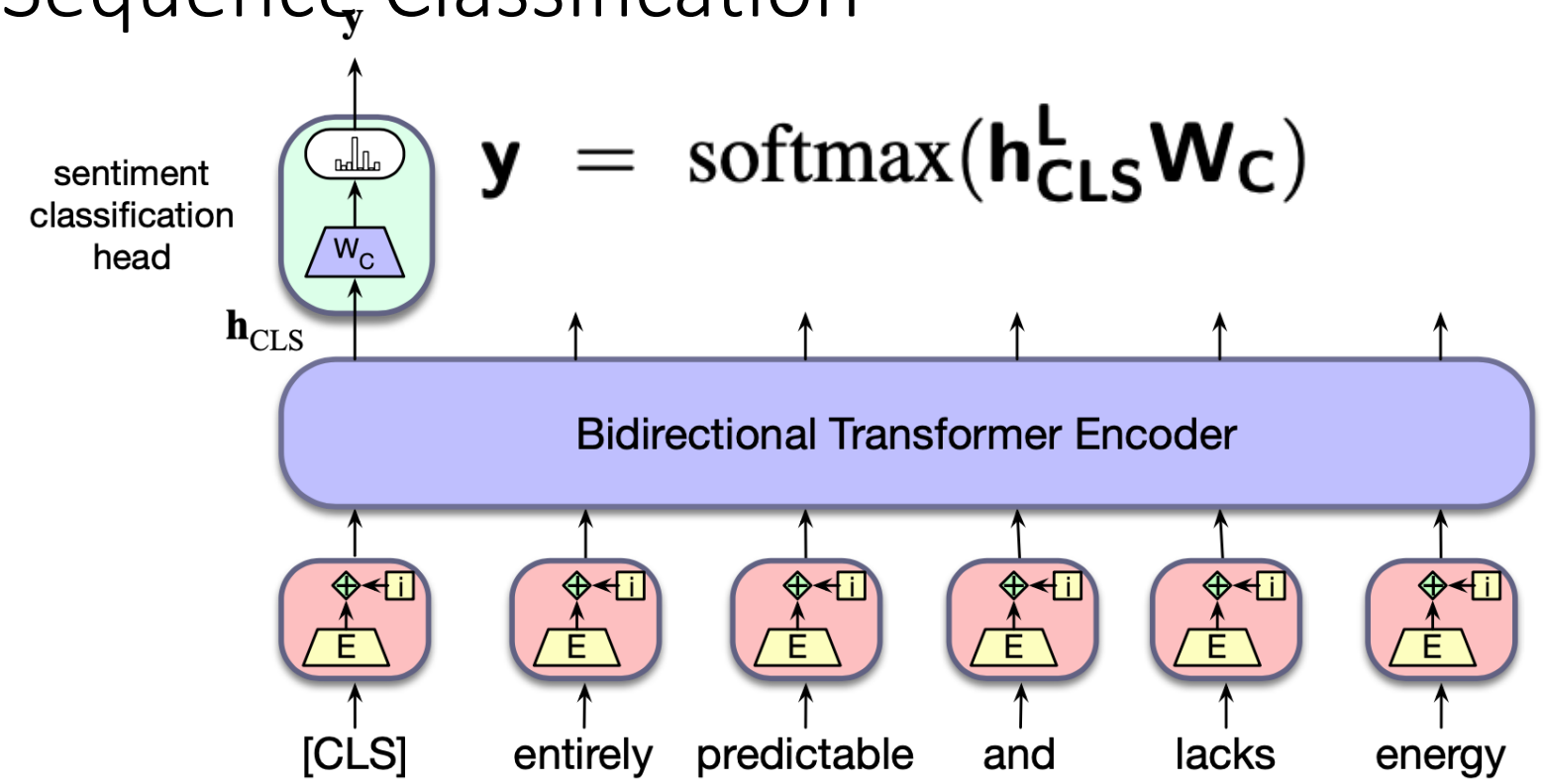
# Sequence Classification

- Nella sequence classification (ad es. la sentiment analysis) si utilizza spesso un *sentence embedding* per consentire la classificazione

- In BERT il token `[CLS]` gioca il ruolo di sentence embedding che è l'input della *testa di classificazione*

- Si esegue il fine-tuning di un layer denso $\mathbf{W}_c$ con dati etichettati e cross-entropy loss

# Sequence Classification



# Classificazione di coppie di frasi

- Riconoscere la relazione tra coppie di frasi: entailment logico, coerenza del discorso, …

- Il fine-tuning è lo stesso dell'addestramento NSP

  - Coppie di frasi vengono presentate al modello usando `[SEP]`
  - Il token `[CLS]` è addestrato con le etichette di classe

# Classificazione di coppie di frasi

- Esempio: textual entailment con il corpus MultiNLI
  - `Neutral`
    - a: Jon walked back to the town to the smithy.
    - b: Jon traveled back to his hometown.
  - `Contradicts`
    - a: Tourist Information offices can be very helpful.
    - b: Tourist Information offices are never of any help.
  - `Entails`
    - a: I'm confused.
    - b: Not all of it is very clear to me.

# Sequence Labelling

- Esempio: Named Entity Recognition con schema BIO per le etichette

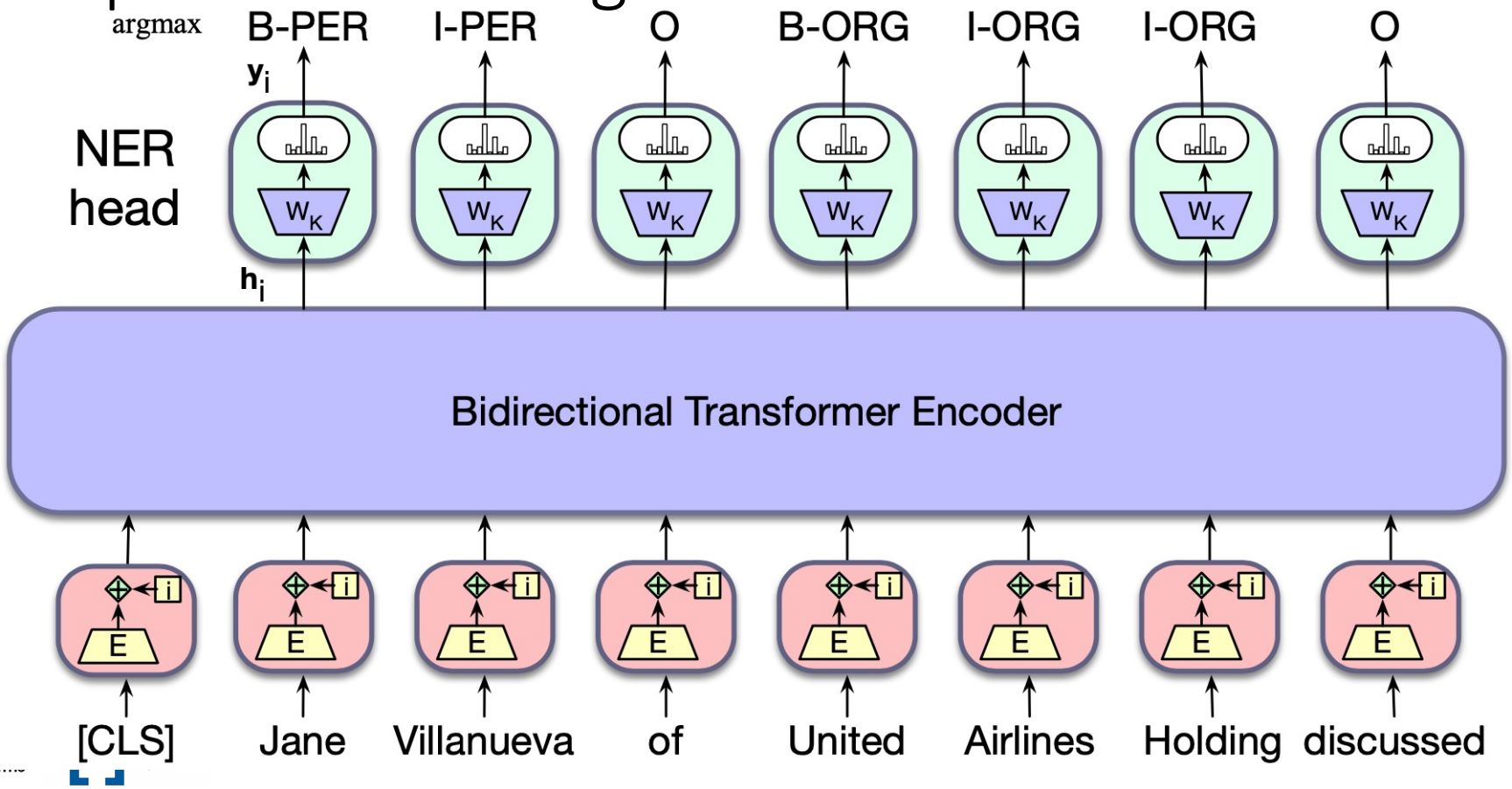
- Si addestra un layer denso $\mathbf{W_K}$ sulle uscite separate di ogni token

    - $\mathbf{W_K}$ ha *k* colonne, una per ogni tag
    - Per selezionare il tag predetto si usa l'*argmax*

# Sequence Labelling

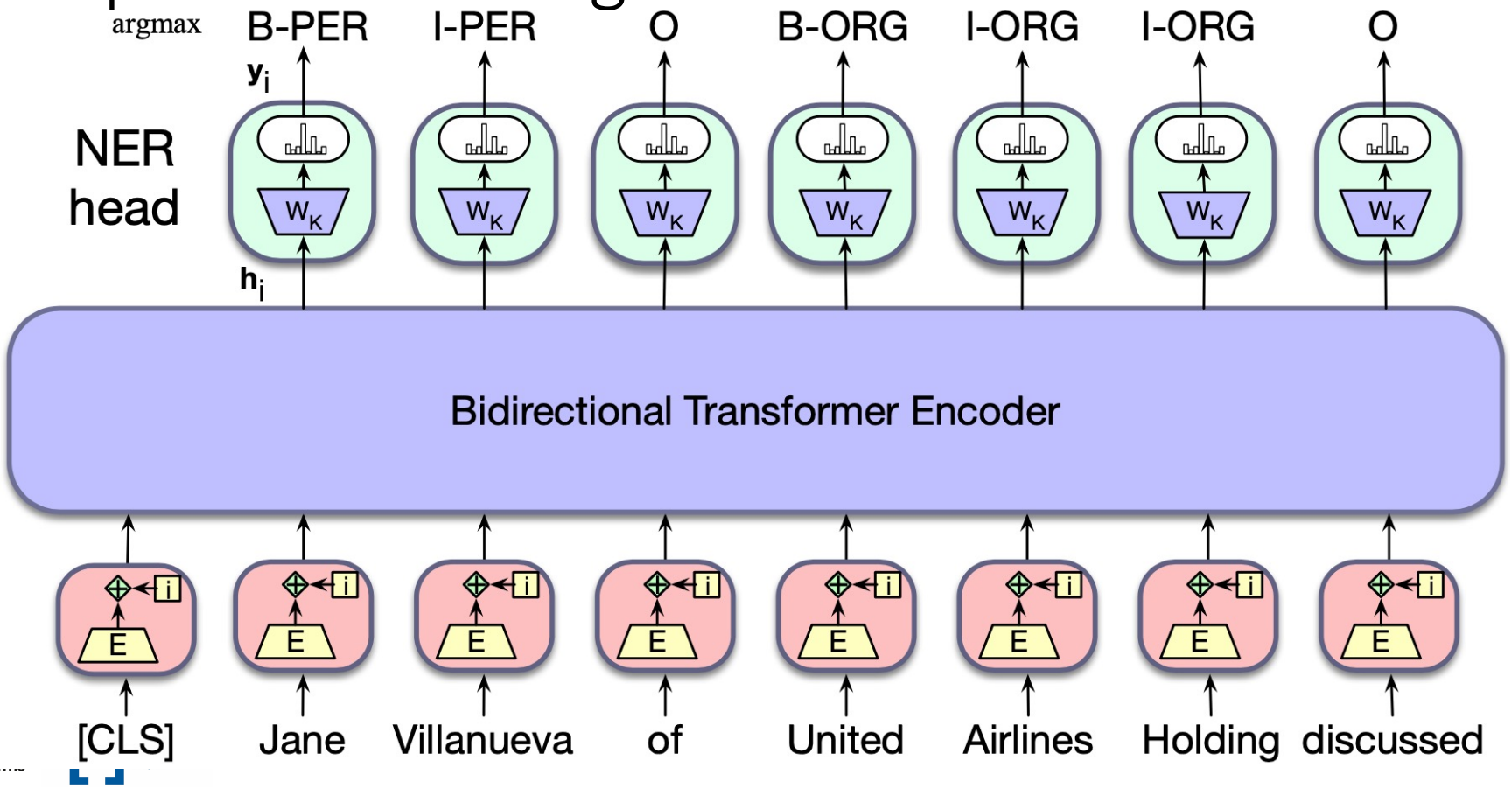
$$\mathbf{y_i} = \text{softmax}(\mathbf{h_i^L W_K})$$

$$\mathbf{t_i} = \text{argmax}_k(\mathbf{y}_i)$$



# Sequence Labelling

*Si può aggiungere un layer CRF sui valori $y_i$ per prendere in considerazione le transizioni globali a livello di tag*



# Sequence Labelling

- Un problema: tokenizzazione subword con WordPiece
- Esempio:

[LOC **Mt. Sanitas** ] is in [LOC **Sunshine Canyon**] .

*Mt. Sanitas is in Sunshine Canyon .*

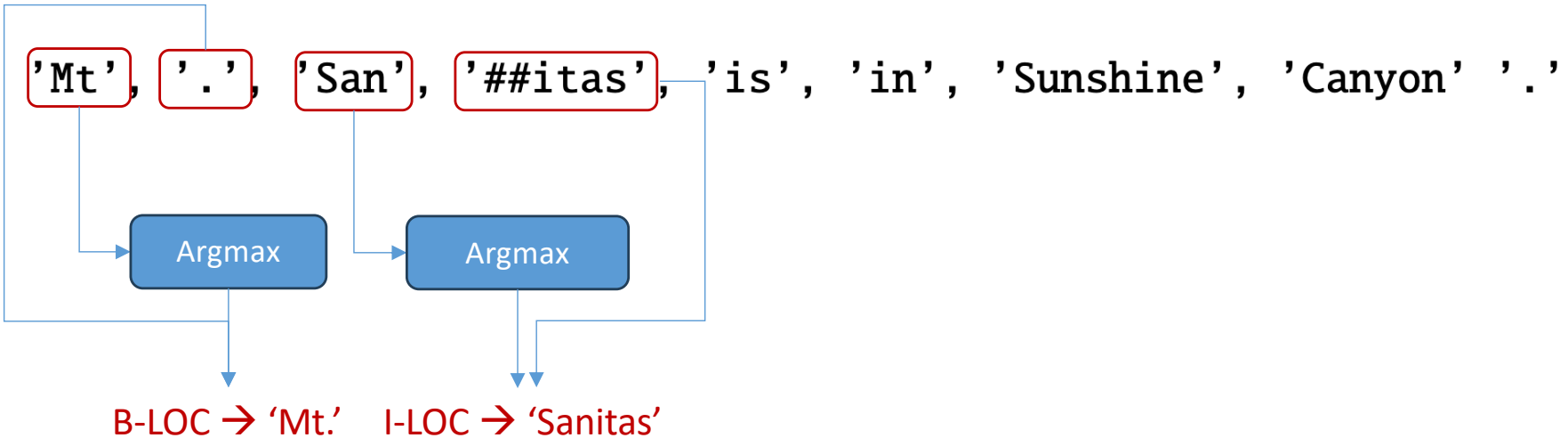
B-LOC I-LOC O O B-LOC I-LOC O

```
'Mt', '.', 'San', '##itas', 'is', 'in', 'Sunshine', 'Canyon' '.'
```

B-LOC B-LOC I-LOC I-LOC O O B-LOC I-LOC O

# Sequence Labelling

- Decodifica attraverso l'*argmax* del tag assegnato al primo token subword




Università degli Studi di Palermo
dipartimento di ingegneria unipa

CHILAB

# Fine-tuning per applicazioni con span

- Le applicazioni orientate agli span operano nel mezzo tra i task orientati alle sequenze e quelli orientati ai token
    - Identificazione di span di interesse
    - Classificazione di span
    - Determinazione di relazione tra gli span

# Fine-tuning per applicazioni con span

- Consideriamo uno span estratto da una sequenza $x$ composta da $T$ tokens che inizia alla posizione $i \in [1, T]$termina alla posizione $j \in [1, T]$
    - Ci sono $T(T - 1)/2$ span possibili in $x$
    - Possiamo selezionare una lunghezza massima $L$
    - $S(x)$ l'insieme degli span legali

# Fine-tuning per applicazioni con span

- Molti approcci per il fine-tuning

- Tutti utilizzano embedding che includono sia la rappresentazione delle estremità sia quella dello span

  - Boundary: la coppia $\langle \mathbf{h}_i, \mathbf{h}_j \rangle$ ovvero si può addestrare una FFN:
    - $s_i = \text{FFN}(\mathbf{h}_i)$
    - $e_j = \text{FFN}(\mathbf{h}_j)$

# Fine-tuning per applicazioni con span

- Molti approcci per il fine-tuning

- Tutti utilizzano embedding che includono sia la rappresentazione delle estremità sia quella dello span

  - Span: $\mathbf{g}_{ij}$ = avg($\mathbf{h}_i$,…,$\mathbf{h}_j$) o meglio la *self-attention*:
  - $\mathbf{g}_{ij}$ = SelfAttention($\mathbf{h}_{i:j}$)

# Fine-tuning per applicazioni con span

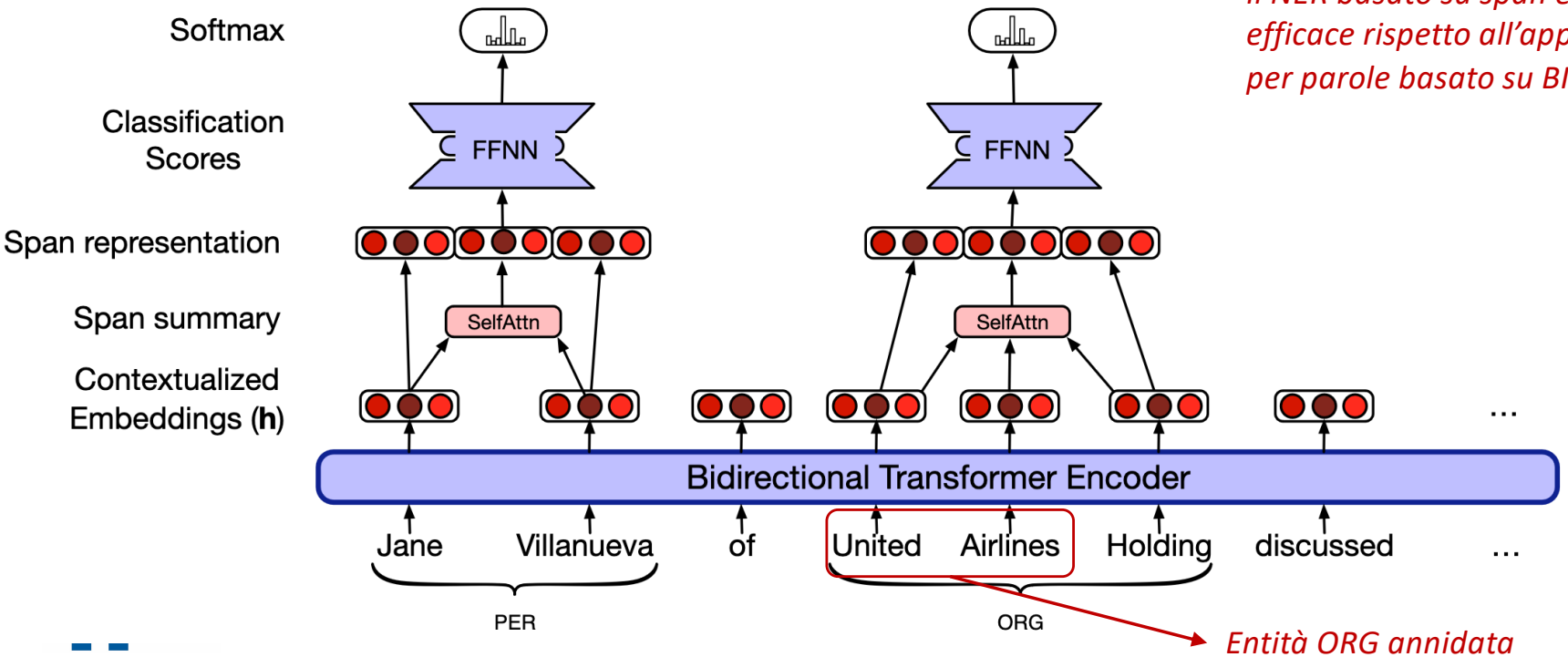
- Molti approcci per il fine-tuning

- Tutti utilizzano embedding che includono sia la rappresentazione delle estremità sia quella dello span

    - Embedding: $\mathbf{emb}_{ij} = \text{Concat}(\mathbf{h}_i, \mathbf{g}_{ij}, \mathbf{h}_j)$

# Fine-tuning per applicazioni con span

$$y_{ij} = \text{softmax}(\text{FFN}(\mathbf{emb}_{ij}))$$




*Il NER basato su span è più efficace rispetto all'approccio per parole basato su BIO.*

# Varianti di BERT

# Multilingual BERT

- Addestrato su 104 lingue contemporaneamente per costruire una rappresentazione condivisa del significato di una parola in ognuna di esse
- Vocabolario di 110K token condivisi per WordPiece
- Non il migliore nei task cross-linguistici

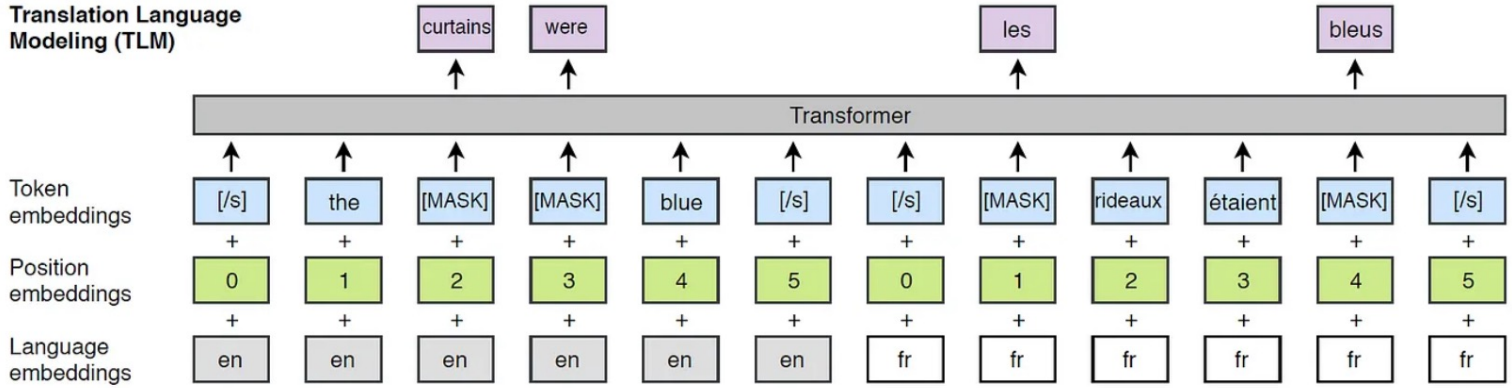
# RoBERTa

- Sviluppato da Facebook (ora Meta)
- Addestrato su cinque corpora inglesi (160GB di testo non compresso)
- Addestramento efficace utilizzando *dynamic masking*
    - Il mascheramento dei token cambia ad ogni sequenza di input
    - BERT esegue un solo masking in fase di preprocessing

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., ... & Stoyanov, V. (2019). RoBERTa: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

# XLM (croX lingual Language Model)

- Sviluppato da Facebook (pre-training su 64 GPUs)
- Tecnica del *Translation Language Modelling* (TLM)




Lample, G., & Conneau, A. (2019). Cross-lingual language model pretraining. *arXiv preprint arXiv:1901.07291*.

Università degli Studi di Palermo

dipartimento di ingegneria unipa

CHILAB

# XLM-RoBERTa

- Sviluppato da Facebook
- Ancora più potere computazionale (pre-training su 500 GPUs) e ancora più dati
- Non utilizza i language embedding per abilitare un veloce switch da una lingua all'altra

Conneau, A., Khandelwal, K., Goyal, N., Chaudhary, V., Wenzek, G., Guzmán, F., ... & Stoyanov, V. (2019). Unsupervised cross-lingual representation learning at scale. *arXiv preprint arXiv:1911.02116*.