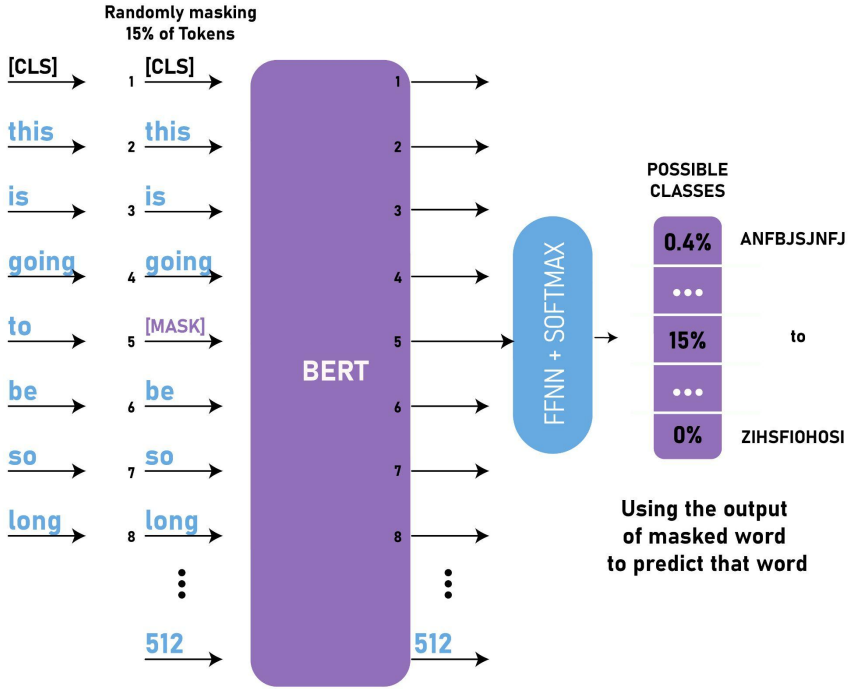
Anatomia degli LLMs

3. Pipeline di un Transformer

Alessio Miaschi
ItaliaNLP Lab, Istituto di Linguistica Computazionale (CNR-ILC), Pisa
alessio.miaschi@ilc.cnr.it
https://alemiaschi.github.io/
http://www.italianlp.it/alessio-miaschi/

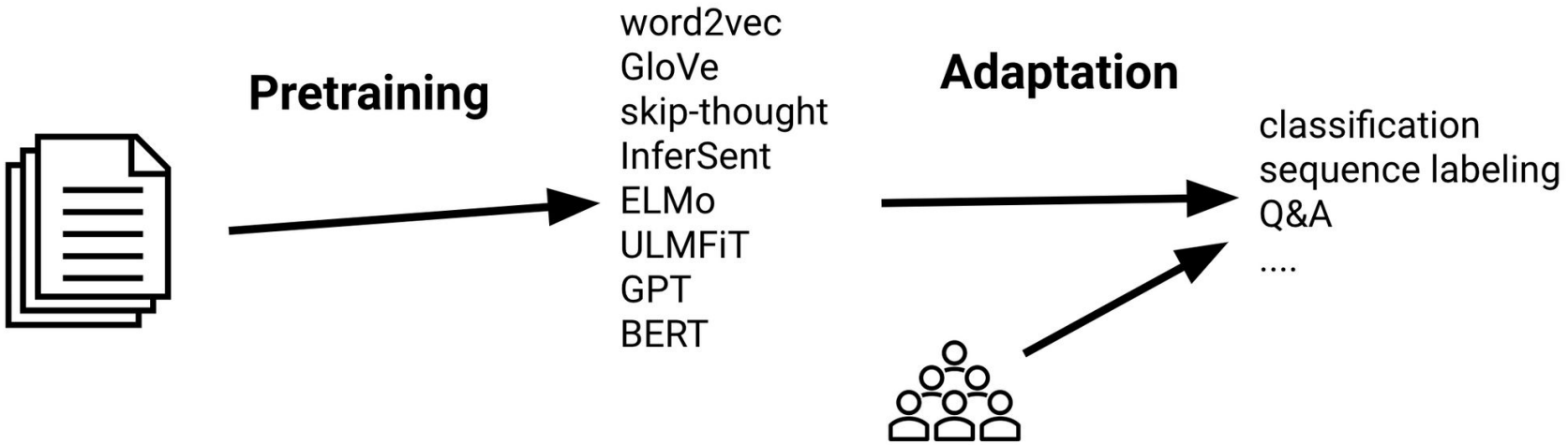
BERT (Devlin et al., 2019)

- Encoder Transformer model (12/24 layers)
- Addestrato per approssimare la funzione di **Masked Language Modeling** (**MLM**)
- Il modello può poi essere ri-addestrato (fine-tuning) per risolvere svariati task di NLP:
  - Sentiment analysis;
  - Question answering;
  - Textual entailment;
  - etc.



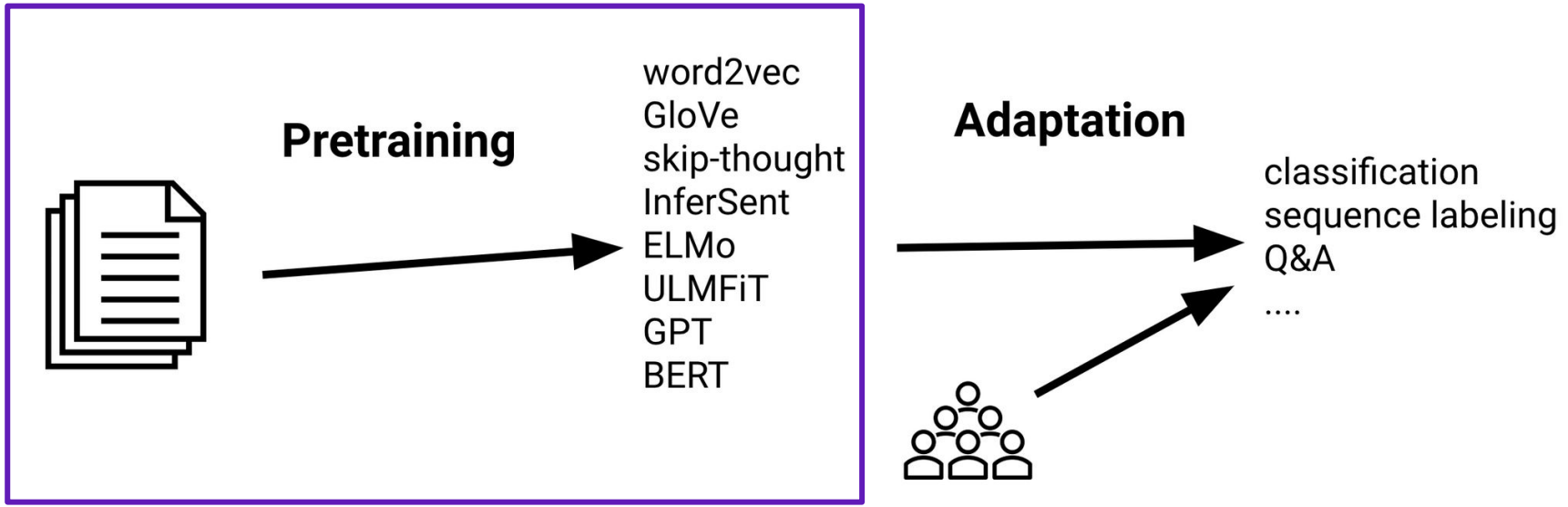
BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding (Devlin et al., 2019), https://aclanthology.org/N19-1423.pdf

Transfer Learning



The State of Transfer Learning in NLP: **https://ruder.io/state-of-transfer-learning-in-nlp/**

Transfer Learning



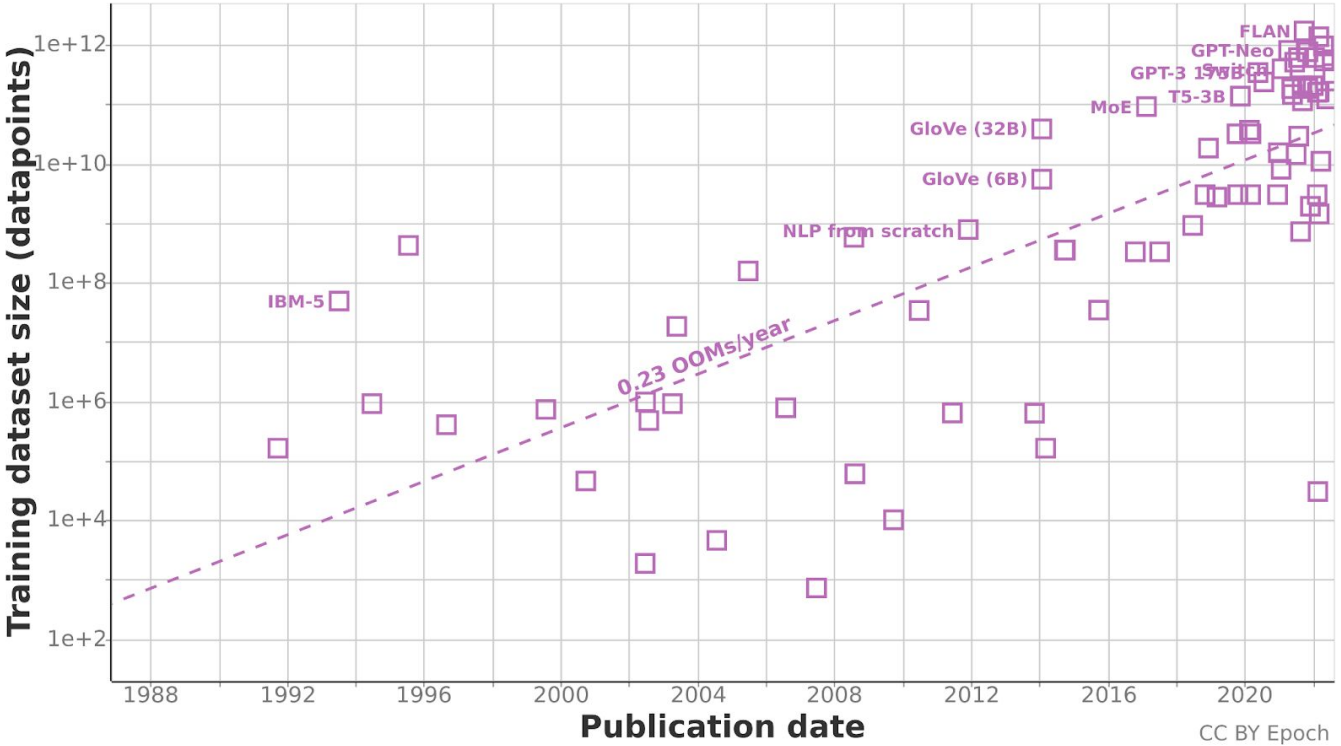
The State of Transfer Learning in NLP: https://ruder.io/state-of-transfer-learning-in-nlp/

Pre-training

- Durante la fase di "*Pre-training*", il modello viene addestrato in maniera unsupervised (e.g. LM, MLM) su una grande quantità di dati grezzi

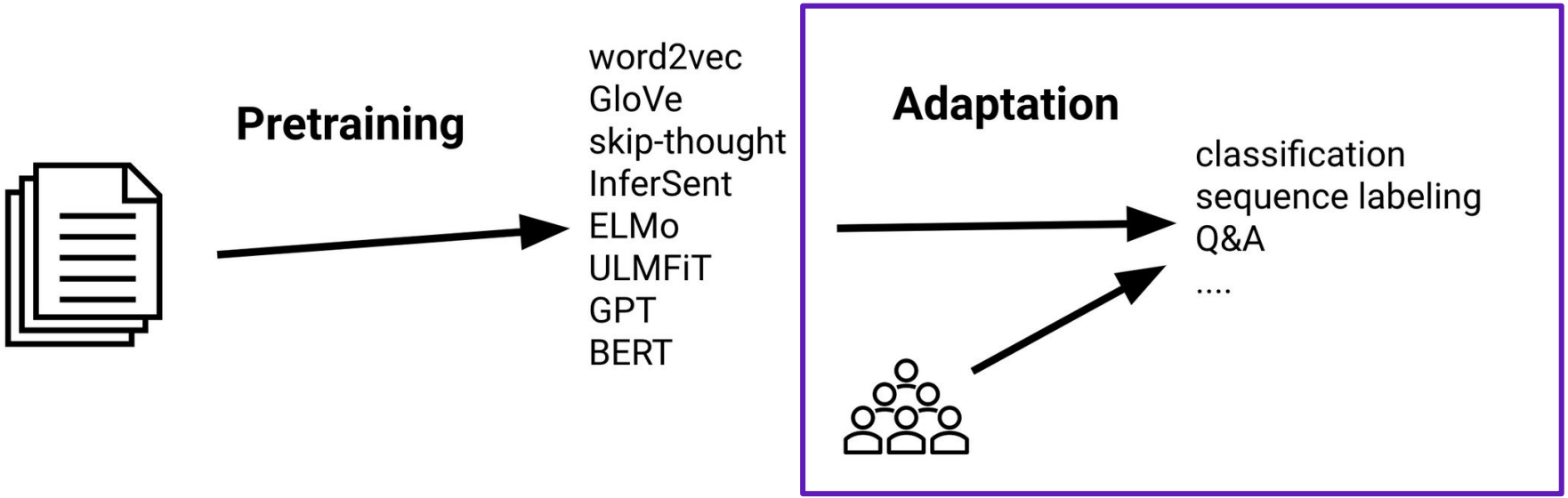
- Alcuni esempi:
  - **Training di BERT**: BookCorpus (800M di parole) e Wikipedia Inglese (2500M di parole)
  - **Training di GPT-3**: CommonCrawl + WebText2 + Books1 + Books2 + Wikipedia (circa 500B di parole)

Pre-training



Fonte: https://www.lesswrong.com/posts/asqDCb9XzXnLjSfgL/trends-in-training-dataset-sizes

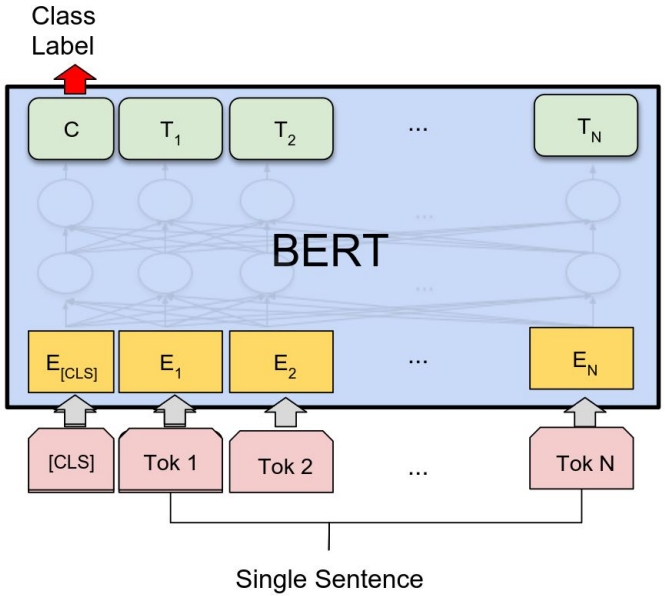
Transfer Learning



The State of Transfer Learning in NLP: https://ruder.io/state-of-transfer-learning-in-nlp/

Fine-tuning (Adaptation)

- Durante la fase di “*Fine-tuning*”, si va a specializzare il modello su un determinato task
  - In altri termini, si prende il modello pre-trainato e si continua l’addestramento su un nuovo dataset e modificando la sua funzione obiettivo (e.g. sentiment analysis, textual entailment, sentence complexity, etc.)



(b) Single Sentence Classification Tasks: SST-2, CoLA

Prompting → Large Language Models (LLMs)

- Negli ultimi anni, lo sviluppo dei NLMs si è spostato verso la creazione di modelli generativi:
  - Scopo principale: considerarsi qualsiasi task (e.g. classificazione, translation, question answering, etc) come task di **generazione**

Prompting → Large Language Models (LLMs)

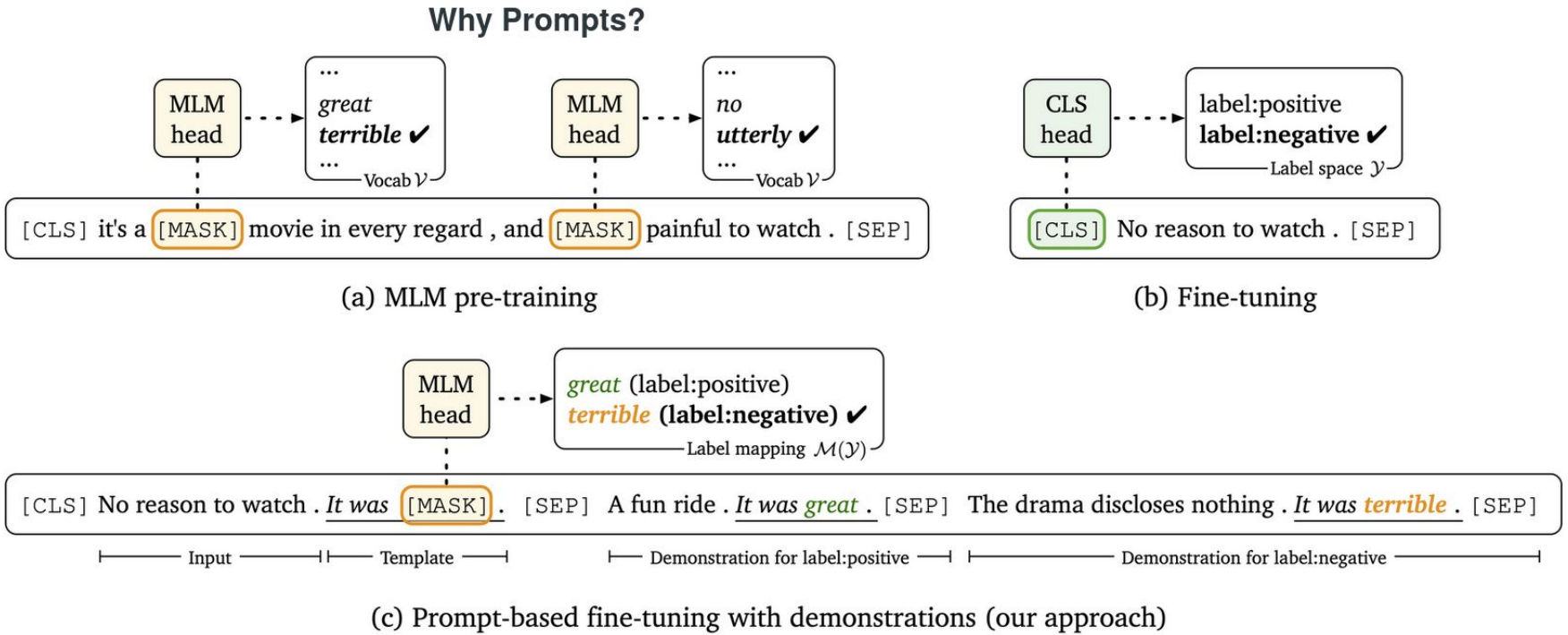
- Negli ultimi anni, lo sviluppo dei NLMs si è spostato verso la creazione di modelli generativi:
  - Scopo principale: considerarsi qualsiasi task (e.g. classificazione, translation, question answering, etc) come task di **generazione**

Prompting

"A prompt is a piece of text inserted in the input examples, so that the original task can be formulated as a (masked) language modeling problem."

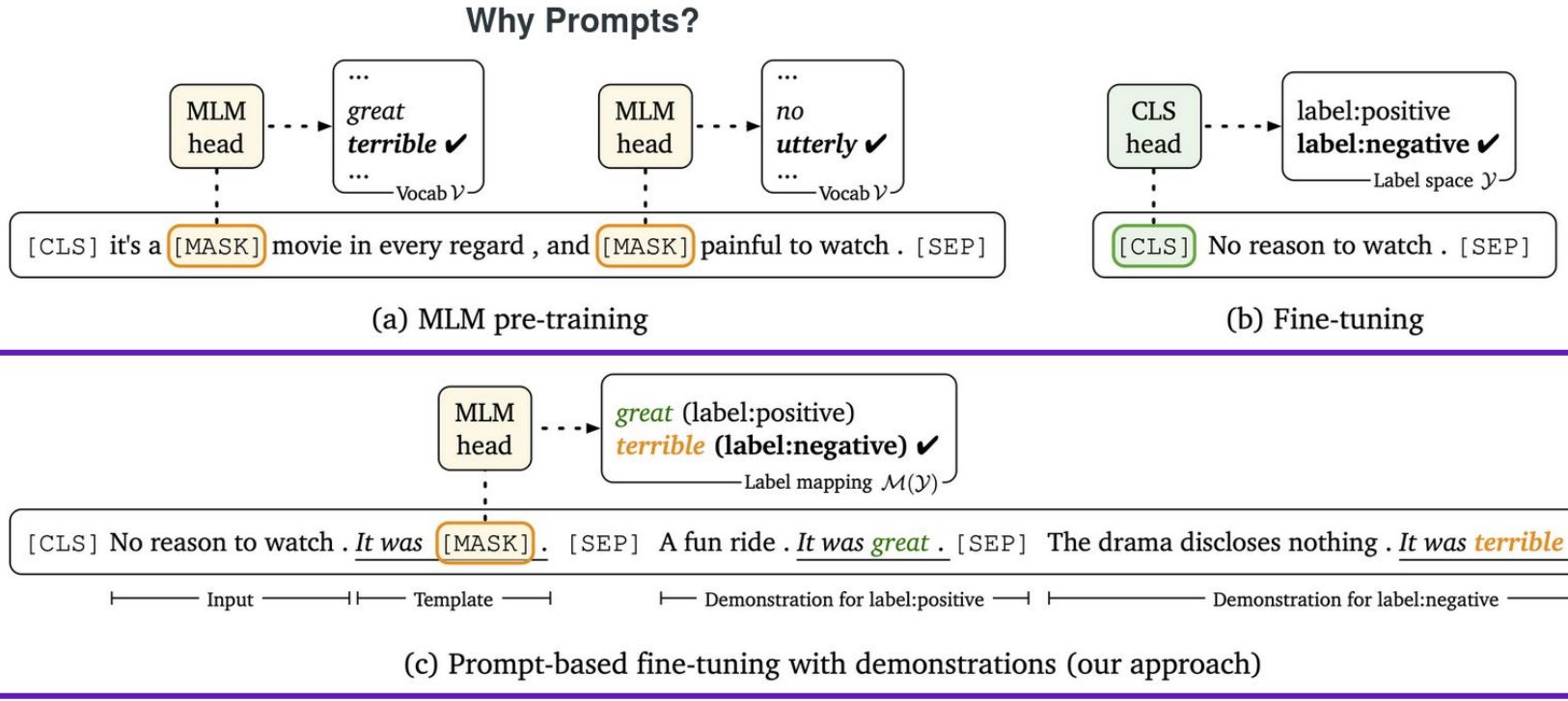
(Prompting: Better Ways of Using Language Models for NLP Tasks, The Gradient)

Prompting → Large Language Models (LLMs)



Fonte: [Prompting: Better Ways of Using Language Models for NLP Tasks, The Gradient]

Prompting → Large Language Models (LLMs)



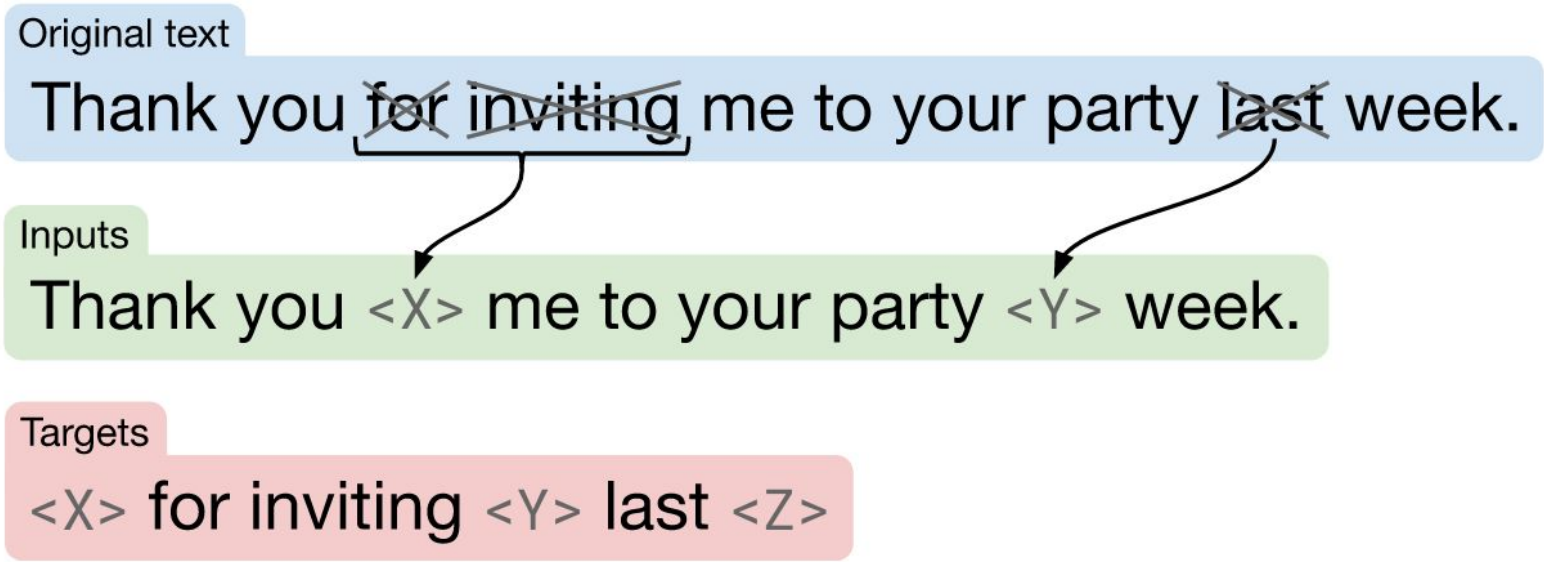
Fonte: [Prompting: Better Ways of Using Language Models for NLP Tasks, The Gradient]

T5 (Raffel et al., 2020)

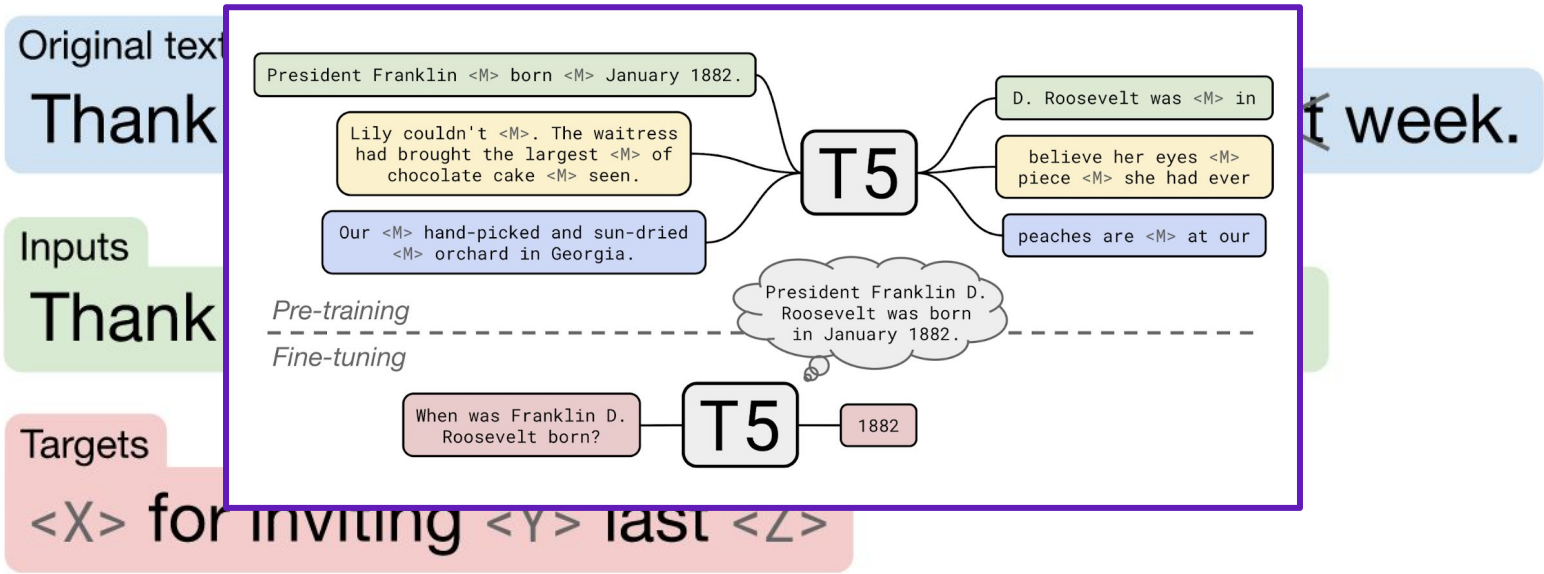
- Encoder-Decoder Transformer model
- Pensato con l'intenzione di riformulare tutti i task di NLP in un formato di tipo "text-to-text", dove input e output sono quindi sempre stringhe di testo

Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer (Raffel et al., 2019), https://dl.acm.org/doi/pdf/10.5555/3455716.3455856

T5 (Raffel et al., 2020) Pre-training

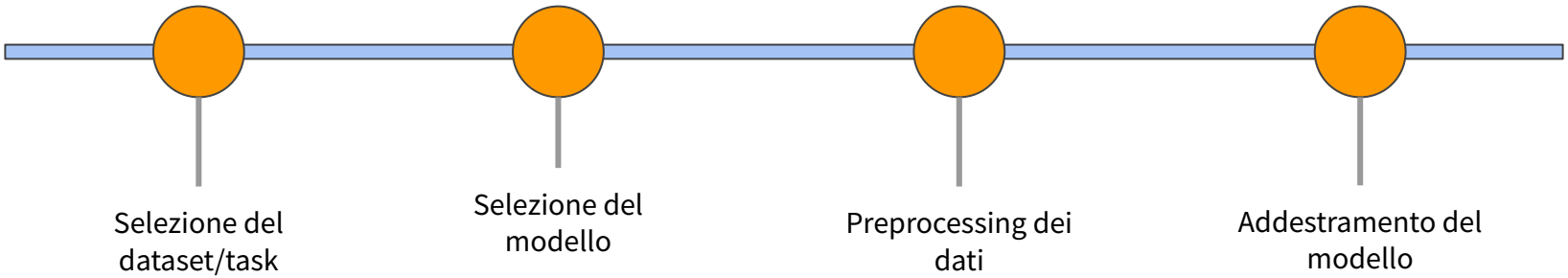


T5 (Raffel et al., 2020) Pre-training

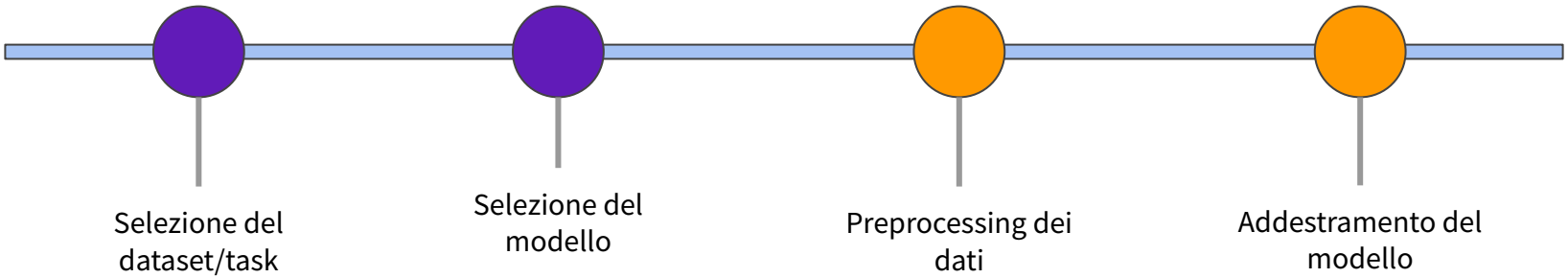


Pipeline di un Transformer

Pipeline di un Transformer Model



Pipeline di un Transformer Model



Definizione del dataset/task e del modello

- È importante avere prima definito il dataset/task su cui addestrare il modello, poiché la scelta del dato/task influenzerà la scelta del modello

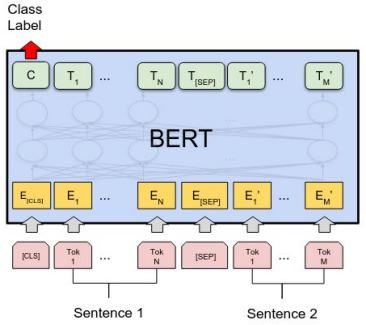
- Alcuni esempi:
  - Generazione (e.g. Text Summarization) → Generative Model (e.g. GPT, T5);
  - Classificazione (e.g. Sentiment Analysis) → Modello per la classificazione (e.g. BERT, RoBERTa)
    - Classificazione binaria, multi-classe, regressione?

Una tassonomia dei Task

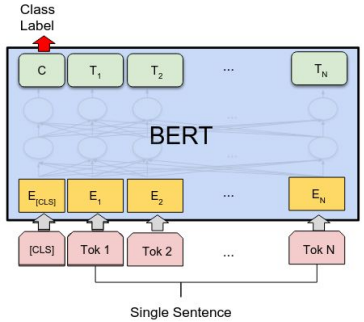
- Text Classification
  - E.g. sentiment analysis, topic classification
- Token Classification
  - E.g. Named-entity Recognition
- Sentence/Document Similarity
- Question Answering

Una tassonomia dei Task

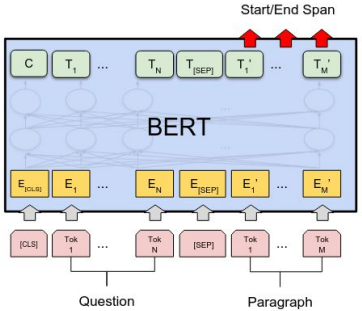
- Text Classification
  - E.g. sentiment analysis, topic classification
- Token Classification
  - E.g. Named-entity Recognition
- Sentence/Document Similarity
- Question Answering



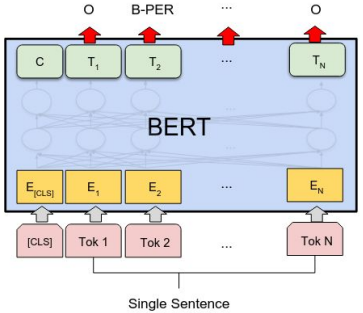
(a) Sentence Pair Classification Tasks: MNLI, QQP, QNLI, STS-B, MRPC, RTE, SWAG



(b) Single Sentence Classification Tasks: SST-2, CoLA



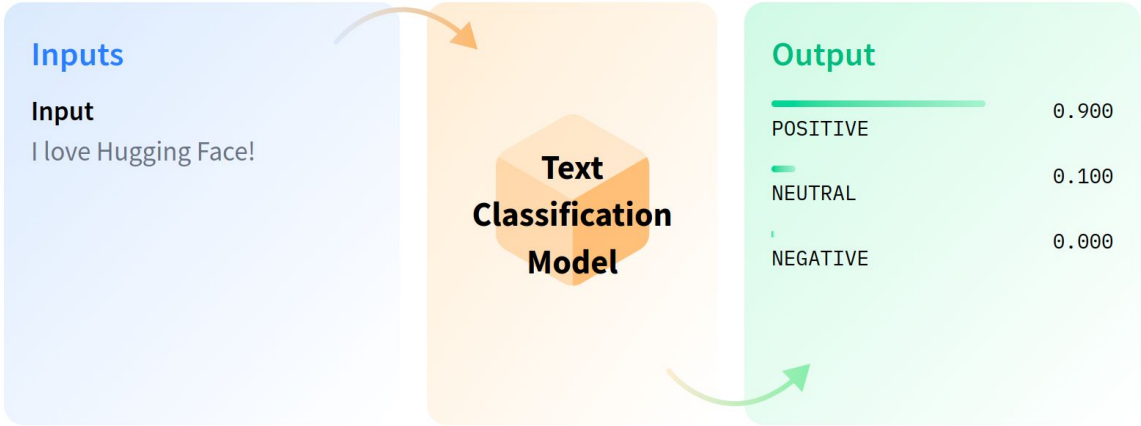
(c) Question Answering Tasks: SQuAD v1.1



(d) Single Sentence Tagging Tasks: CoNLL-2003 NER

Text Classification

- La **Text Classification** è il compito di assegnare un'etichetta o una classe a un dato testo
- Alcuni casi d'uso sono la sentiment analysis, la natural language inference (NLI) e la valutazione della correttezza grammaticale



https://huggingface.co/tasks/text-classification

Text Classification

- Sentiment Analysis

Example

Text: Read the book, forget the movie!
Label: Negative

- Natural Language Inference

Example

Text: A soccer game with multiple males playing.
Hypothesis: Some men are playing sport.
Label: Entailment

Text Classification

- Sentiment Analysis

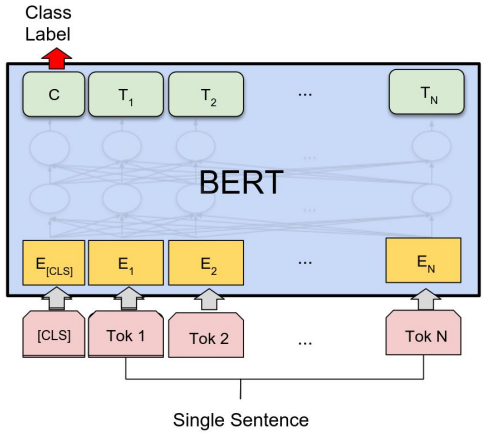
Example

Text: Read the book, forget the movie!
Label: Negative

- Natural Language Inference

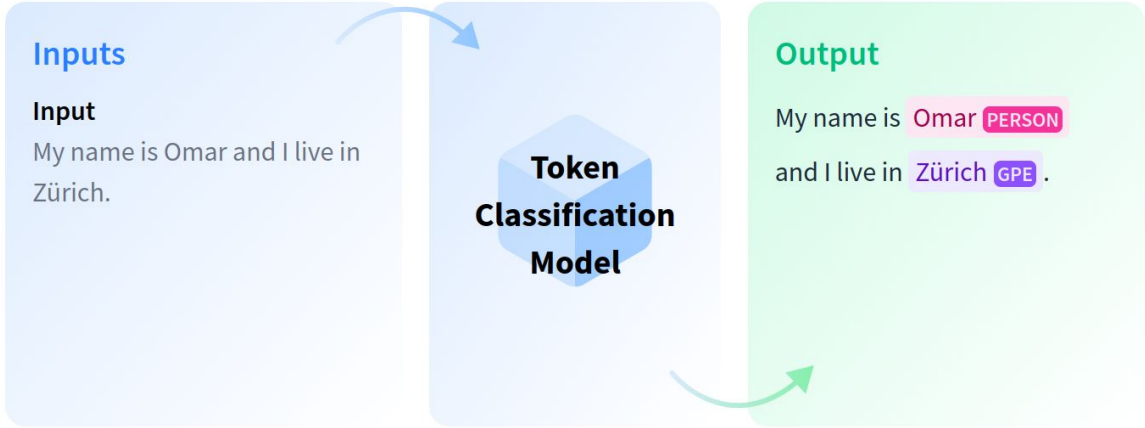
Example

Text: A soccer game with multiple males playing.
Hypothesis: Some men are playing sport.
Label: Entailment



Token Classification

- La **Token Classification** è il task in cui viene assegnata un'etichetta ad alcuni token di un testo.
- Tra i vari task di Token Classification vi sono la Named Entity Recognition (NER) e il Part-of-Speech (PoS) tagging



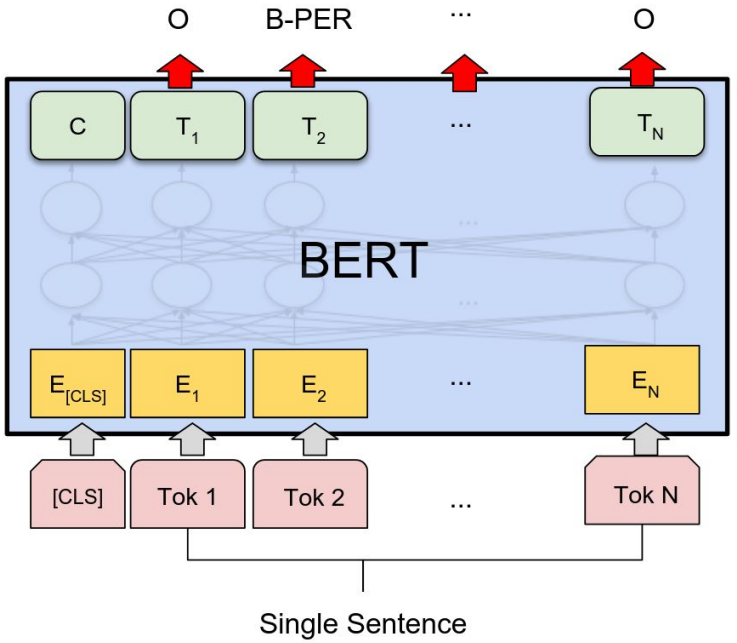
https://huggingface.co/tasks/token-classification

NER e POS Tagging

Token	POS tag	NER tag
The	DT	O
FAA	NNP	B-ORG
is	VBZ	O
headquartered	VBN	O
in	IN	O
Washington	NNP	B-LOC
DC	NNP	I-LOC

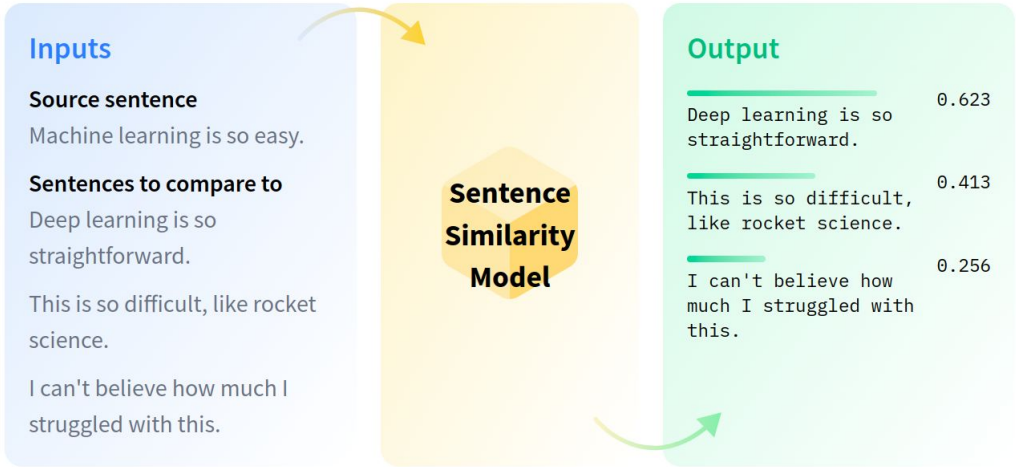
NER e POS Tagging

Token	POS tag	NER tag
The	DT	O
FAA	NNP	B-ORG
is	VBZ	O
headquartered	VBN	O
in	IN	O
Washington	NNP	B-LOC
DC	NNP	I-LOC



Sentence/Document Similarity

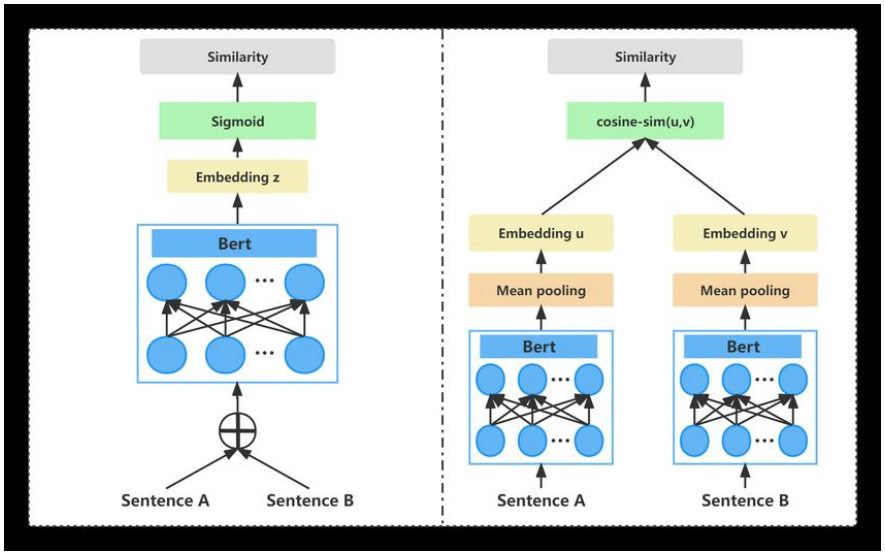
- La **sentence similarity** è il task di determinare la somiglianza tra due testi.
- I modelli/sistemi di sentence similarity convertono i testi in ingresso in vettori (embeddings) che catturano le informazioni semantiche e calcolano quanto sono vicini (simili) tra loro



https://huggingface.co/tasks/sentence-similarity

Sentence/Document Similarity

Sentence/Document Similarity

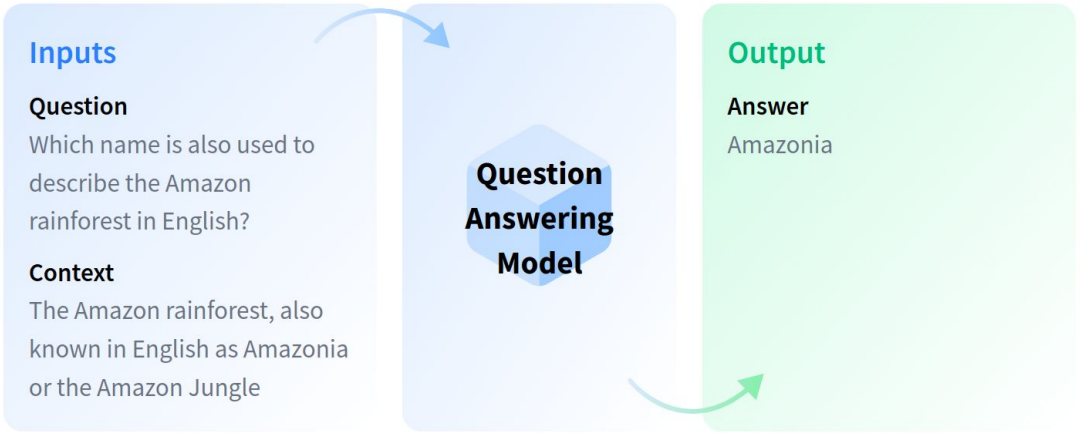


BERT SentenceBERT

https://huggingface.co/sentence-transformers

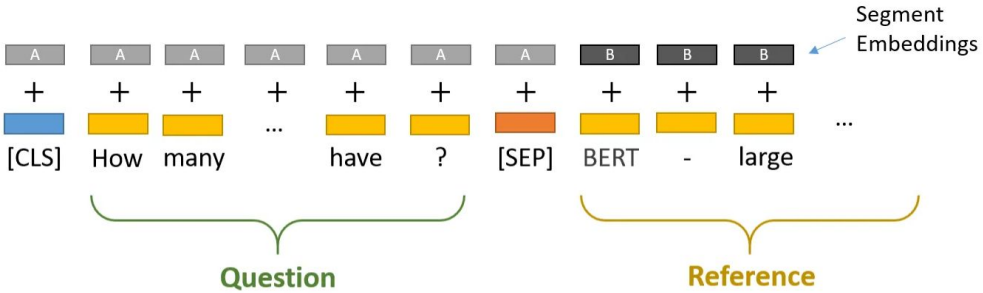
Question Answering

- Il **Question Answering** (**QA**) è il task di recupero di una risposta a una domanda da un testo dato
- **Extractive QA**: Il modello estrae la risposta dal contesto (e.g. testo, tabella, ecc). Questo problema viene solitamente risolto con BERT-based models



https://huggingface.co/tasks/question-answering

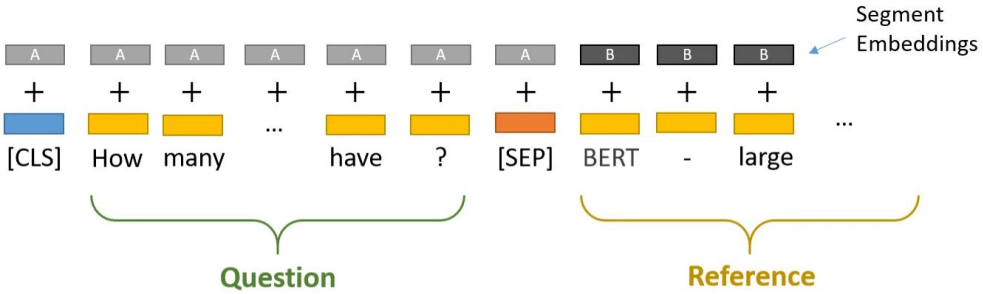
Question Answering



Question: How many parameters does BERT-large have?

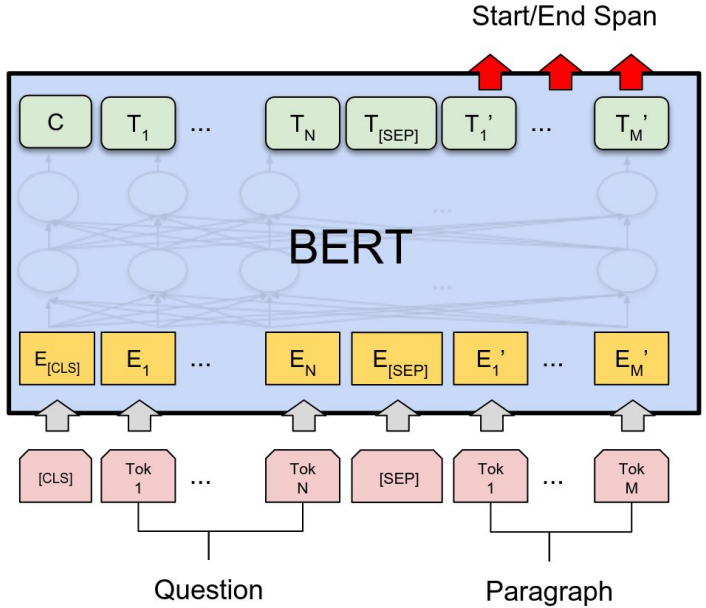
Reference Text: BERT-large is really big... it has 24 layers and an embedding size of 1,024, for a total of 340M parameters! Altogether it is 1.34GB, so expect it to take a couple minutes to download to your Colab instance.

Question Answering

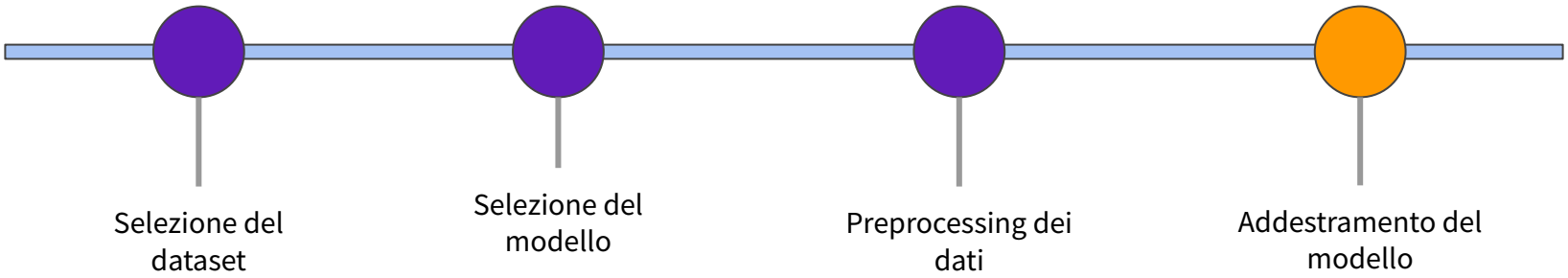


Question: How many parameters does BERT-large have?

Reference Text: BERT-large is really big... it has 24 layers and an embedding size of 1,024, for a total of 340M parameters! Altogether it is 1.34GB, so expect it to take a couple minutes to download to your Colab instance.



Pipeline di un Transformer Model



Preprocessing dei dati

- Prima di poter passare una sequenza (e.g. frase, documento) ad un NLM, è necessario passare precedentemente da una fase di tokenizzazione

- A seconda della tipologia di modello utilizzato, esistono diversi tokenizzatori in grado di segmentare il testo
  - Byte-Pair Encoding (BPE); WordPiece

- I principi dietro ai tokenizzatori maggiormente utilizzati con i recenti NLM sono:
  - Parole di uso frequente non devono essere suddivise in sottoparole più piccole
  - Parole rare (meno frequenti) devono essere scomposte in sottoparole significative

Byte-Pair Encoding (BPE) Tokenization

- Il **Byte-Pair Encoding** (**BPE**) è stato inizialmente sviluppato come algoritmo per comprimere i testi e poi utilizzato da OpenAI per la tokenizzazione durante il pre-training del primo GPT

- Algoritmo:
  1. Ogni parola viene suddivisa in singoli caratteri
  2. Calcolo della coppia di caratteri adiacenti più frequente nel testo
  3. Unione della coppia in un nuovo “subtoken” da aggiungere al vocabolario
  4. Ripetizione dei passaggi 2-3 finché non si raggiunge il numero desiderato di token

Byte-Pair Encoding (BPE) Tokenization

Training corpus: low low low low low lowest lowest newer newer newer newer newer newer wider wider wider new new

Corpus	
5	l o w __
2	l o w e s t __
6	n e w e r __
3	w i d e r __
2	n e w __

Vocabulary

__, d, e, i, l, n, o, r, s, t, w

https://cs.usm.maine.edu/~behrooz.mansouri/courses/Slides_NLP_23/Natural%20Language%20Processing%20--%20Session%204%20-%20Tokenization%20and%20Stemming.pdf

Byte-Pair Encoding (BPE) Tokenization

Training corpus: low low low low low lowest lowest newer newer newer newer newer newer wider wider wider new new

9 times

Corpus	
5	l o w __
2	l o w e s t __
6	n e w e r __
3	w i d e r __
2	n e w __

Vocabulary

__, d, e, i, l, n, o, r, s, t, w

Vocabulary

__, d, e, i, l, n, o, r, s, t, w, er

https://cs.usm.maine.edu/~behrooz.mansouri/courses/Slides_NLP_23/Natural%20Language%20Processing%20--%20Session%204%20-%20Tokenization%20and%20Stemming.pdf

Byte-Pair Encoding (BPE) Tokenization

Training corpus: low low low low low lowest lowest newer newer newer newer newer newer wider wider wider new new

9 times

Corpus	
5	l o w __
2	l o w e s t __
6	n e w e r __
3	w i d e r __
2	n e w __

Vocabulary

__, d, e, i, l, n, o, r, s, t, w, er

Vocabulary

__, d, e, i, l, n, o, r, s, t, w, er, er__

https://cs.usm.maine.edu/~behrooz.mansouri/courses/Slides_NLP_23/Natural%20Language%20Processing%20--%20Session%204%20-%20Tokenization%20and%20Stemming.pdf

Byte-Pair Encoding (BPE) Tokenization

Training corpus: low low low low low lowest lowest newer newer newer newer newer newer wider wider wider new new

8 times

Corpus	
5	l o w __
2	l o w e s t __
6	n e w e r __
3	w i d e r __
2	n e w __

Vocabulary

__, d, e, i, l, n, o, r, s, t, w, er, er__

Vocabulary

__, d, e, i, l, n, o, r, s, t, w, er, er__, ne

https://cs.usm.maine.edu/~behrooz.mansouri/courses/Slides_NLP_23/Natural%20Language%20Processing%20--%20Session%204%20-%20Tokenization%20and%20Stemming.pdf

Byte-Pair Encoding (BPE) Tokenization

Training corpus: low low low low low lowest lowest newer newer newer newer newer newer wider wider wider new new

8 times

	Corpus
5	l o w __
2	l o w e s t __
6	n e w e r __
3	w i d e r __
2	n e w __

Vocabulary

__, d, e, i, l, n, o, r, s, t, w, er, er__, ne

Vocabulary

__, d, e, i, l, n, o, r, s, t, w, er, er__, ne, new

https://cs.usm.maine.edu/~behrooz.mansouri/courses/Slides_NLP_23/Natural%20Language%20Processing%20--%20Session%204%20-%20Tokenization%20and%20Stemming.pdf

Byte-Pair Encoding (BPE) Tokenization

Corpus

5 l o w __
2 l o w e s t __
6 n e w e r __
3 w i d e r __
2 n e w __

Final Vocabulary

__, d, e, i, l, n, o, r, s, t, w, er, er__, ne, new, lo, low, newer __, low__

Using BPE for tokenization:

Input: newer__ → Tokens: newer__

Input: lower__ → Tokens: low, er__

Merge based on the order we learned:
er → er__ → ne → new → newer__
er → er__ → lo → low

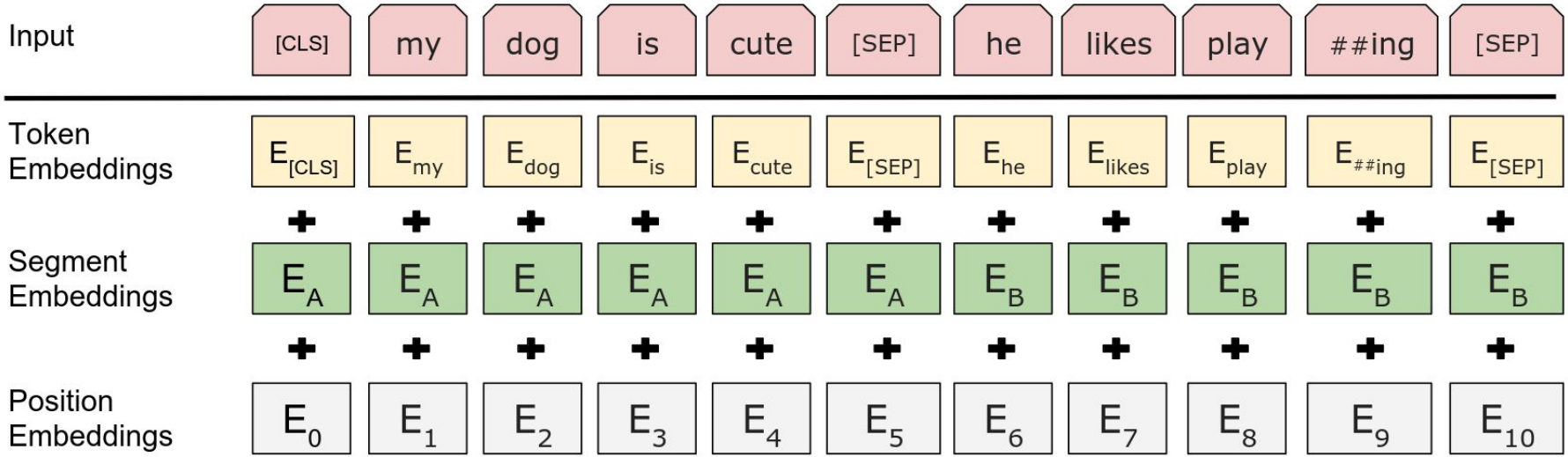
https://cs.usm.maine.edu/~behrooz.mansouri/courses/Slides_NLP_23/Natural%20Language%20Processing%20--%20Session%204%20-%20Tokenization%20and%20Stemming.pdf

WordPiece Tokenizer

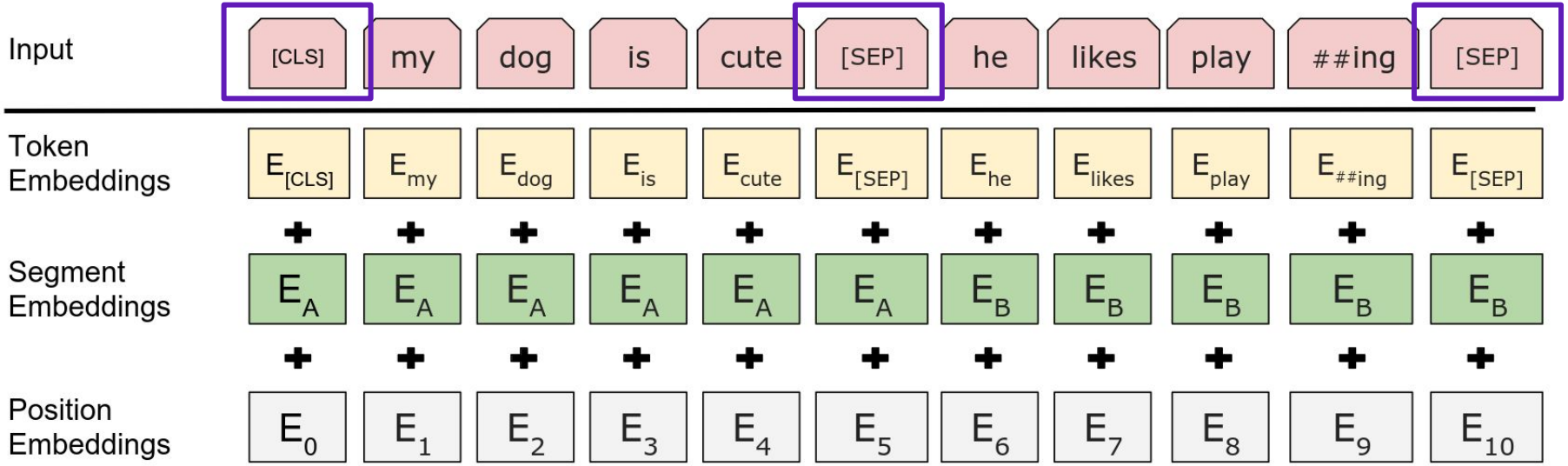
- **WordPiece** è l'algoritmo di tokenizzazione sviluppato da Google per il pretraining di BERT
- Simile a BPE in termini di addestramento, ma la tokenizzazione vera e propria avviene in modo diverso
- Come BPE, ogni parola viene suddivisa in caratteri, aggiungendo un prefisso per identificare tutti i caratteri all'interno della parola
  - **word → w #o #r #d**
- La differenza principale con BPE è il modo in cui vengono selezionate le coppie da unire. Invece di selezionare la coppia più frequente, WordPiece calcola un punteggio:

$$\text{score}=(\text{freq\_of\_pair})/(\text{freq\_of\_first\_element}\times\text{freq\_of\_second\_element})$$

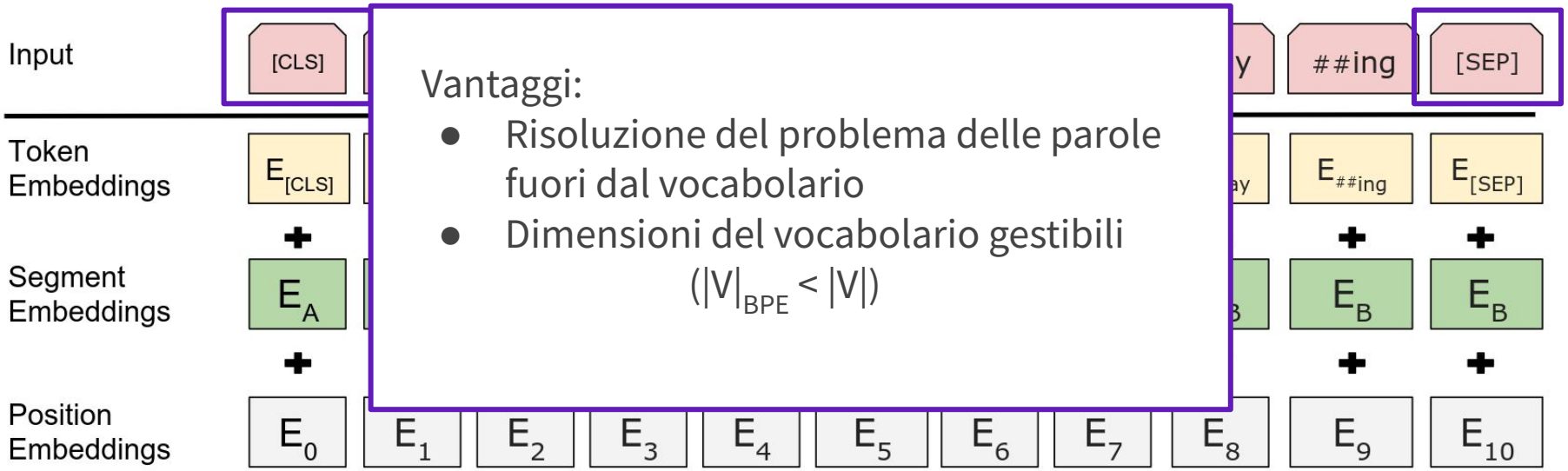
WordPiece Tokenizer



WordPiece Tokenizer



WordPiece Tokenizer

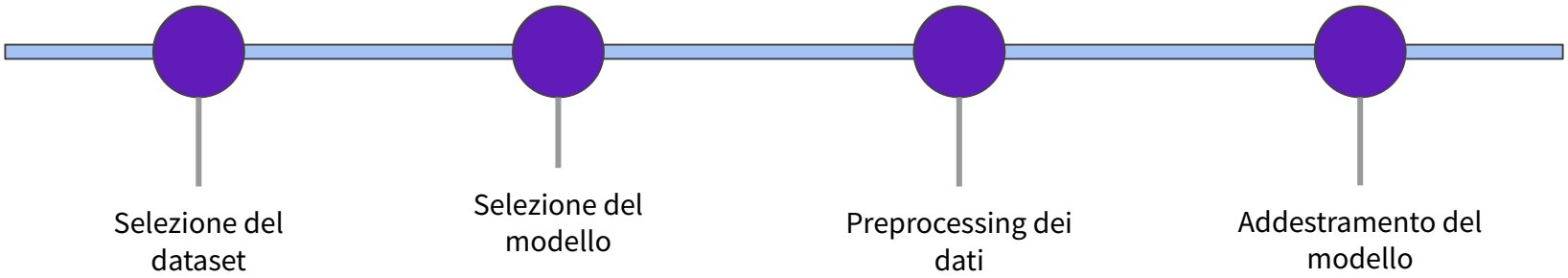


Vantaggi:

- Risoluzione del problema delle parole fuori dal vocabolario
- Dimensioni del vocabolario gestibili

$$(|V|_{BPE} < |V|)$$

Pipeline di un Transformer Model



Addestramento del modello

- Dopo aver tokenizzato il testo, è possibile passarlo in input al modello e iniziare la fase di addestramento:
  - Pre-training: addestramento del modello da zero, usando come funzione obiettivo il task di LM o una sua variante (e.g. MLM) → necessaria grande quantità di dati e di risorse computazionali
  - Fine-tuning: specializzazione del modello su un task, a partire dai pesi già addestrati durante la fase di pre-training

Addestramento del modello

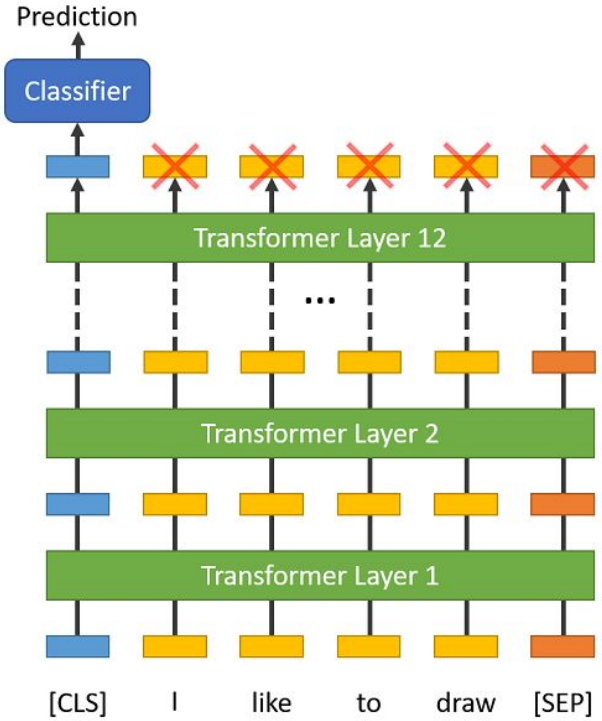
- Dopo aver tokenizzato il testo, è possibile passarlo in input al modello e iniziare la fase di addestramento:
  - Pre-training: addestramento del modello da zero, usando come funzione obiettivo il task di LM o una sua variante (e.g. MLM) → necessaria grande quantità di dati e di risorse computazionali
  - **Fine-tuning: specializzazione del modello su un task, a partire dai pesi già addestrati durante la fase di pre-training**

Addestramento del modello

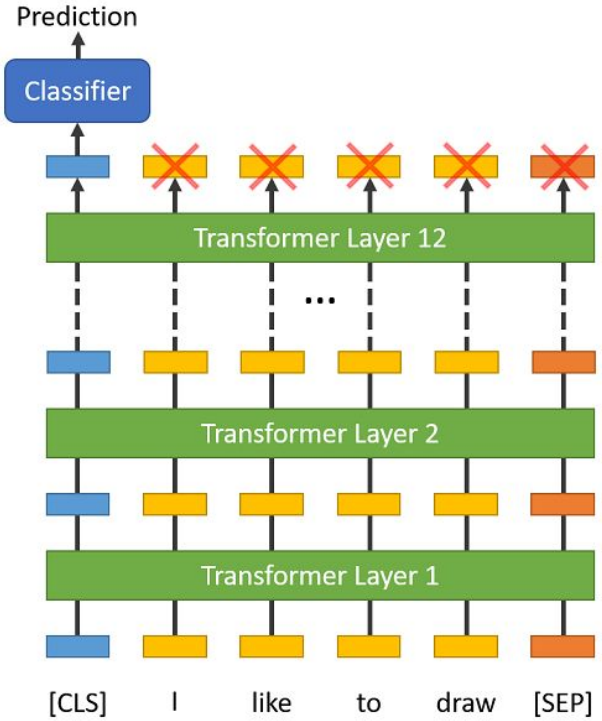
- Dopo aver tokenizzato il testo, è possibile passarlo in input al modello e iniziare la fase di addestramento:
  - Pre-training: addestramento del modello da zero, usando come funzione obiettivo il task di LM o una sua variante (e.g. MLM) → necessaria grande quantità di dati e di risorse computazionali
  - **Fine-tuning: specializzazione del modello su un task, a partire dai pesi già addestrati durante la fase di pre-training**

- Alcune avvertenze:
  - Per addestrare un Transformer model in maniera ottimale è consigliato (se non necessario) ricorrere all'utilizzo di una (o più) **GPU** (Graphics Processing Unit)
  - La complessità del modello dipende anche dalla dimensione delle sequenze in input → +testo = +pesi da addestrare

Addestramento del modello



Addestramento del modello

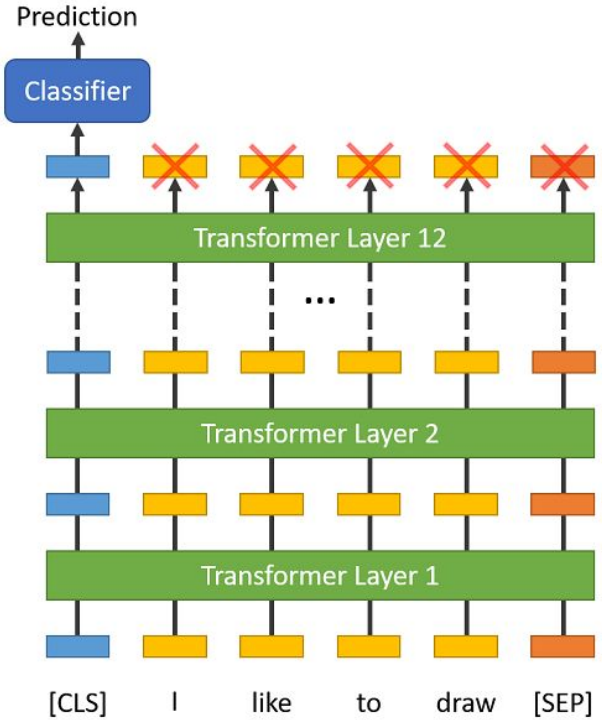


$p^* = (0, \ldots, 0, 1, 0, \ldots)$ distribuzione target

$p = (p_1, \ldots, p_K)$ distribuzione del modello

$$Loss(p^*, p) = -p^* \log(p) = -\sum_{i=1}^{K} p_i^* \log(p_i)$$

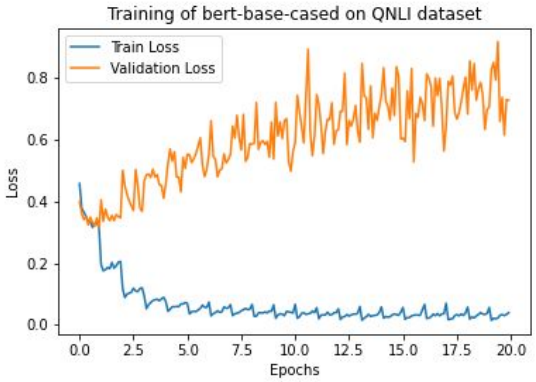
Addestramento del modello



$p^* = (0, \ldots, 0, 1, 0, \ldots)$ distribuzione target

$p = (p_1, \ldots, p_K)$ distribuzione del modello

$$Loss(p^*, p) = -p^* \log(p) = -\sum_{i=1}^{K} p_i^* \log(p_i)$$



La libreria *Transformers* 🤗

- La libreria **Transformers** (di Huggingface) è, ad oggi, la risorsa open source più utilizzata per poter scaricare, modificare e addestrare facilmente i Transformer models

📝 **Natural Language Processing**: text classification, named entity recognition, question answering, language modeling, summarization, translation, multiple choice, and text generation.

🖼️ **Computer Vision**: image classification, object detection, and segmentation.

🗣️ **Audio**: automatic speech recognition and audio classification.

🐙 **Multimodal**: table question answering, optical character recognition, information extraction from scanned documents, video classification, and visual question answering.