Deep Learning

Università Roma Tre
Dipartimento di Ingegneria

Anno Accademico 2022 - 2023

Deep Learning e Natural Language Processing

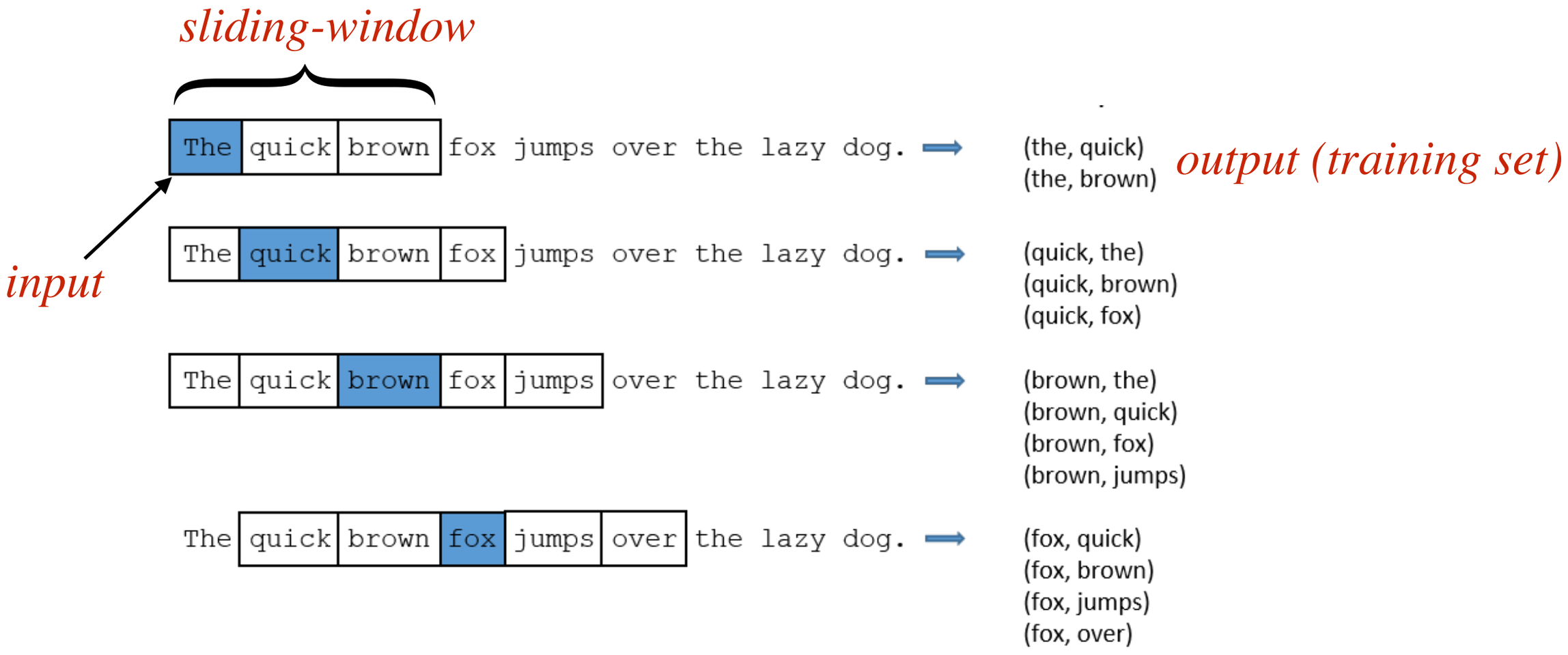
Sommario

- DL e NLP: motivazioni
- word2vec
  - skip-gram
  - CBOW
- GloVe
- fastText
- BERT

DL e NLP

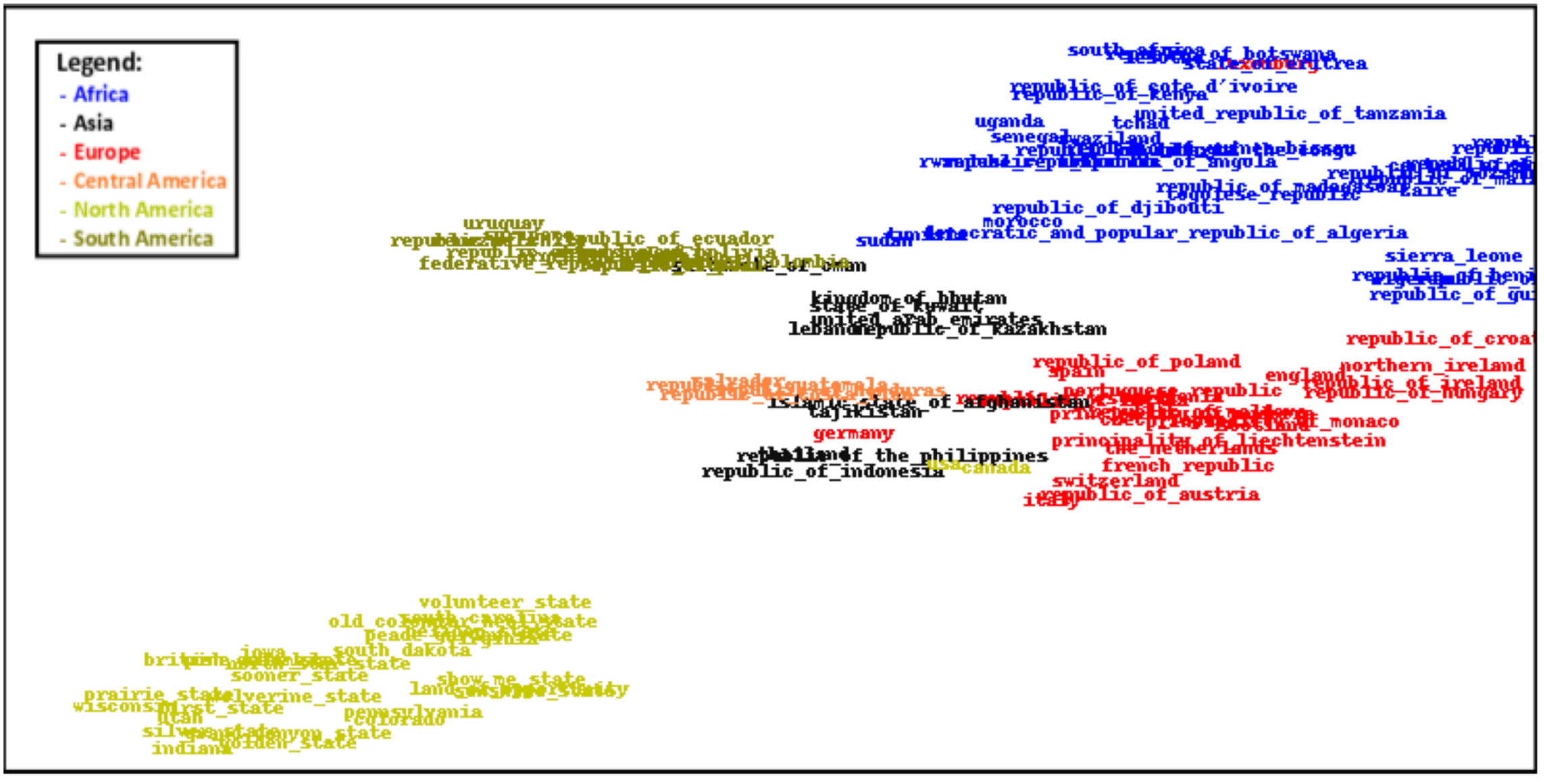
- Durante l'apprendimento di tecniche ML abbiamo spesso bisogno di grosse moli di dati *labelled* per implementare approcci supervisionati.
- Alcune architetture DL hanno la capacità di riconoscere pattern e caratteristiche anche complesse in tali dati, ma dataset adeguati per l'addestramento non sono disponibili.
- Per tale motivo sono stati proposti vari approcci come il *self-supervised learning*, per analizzare dati un modo non supervisionato (*auxiliary task, es.* predire una parte mancante del testo) e costruire rappresentazioni utili per supportare l'apprendimento (tipicamente supervisionato) in task più specifici.
- Avendo un dataset di testo, l'input può essere costruito impiegando singole parole o n-grams formati da lettere, utili per catturare informazioni morfologiche delle parole. L'output è tipicamente una rappresentazione vettoriale associata ad ogni parola (*embedding*), indipendente dal contesto in cui sarà presente in seguito.

Esempio di auxiliary task



Esempio di embeddings

- Embeddings relativi a termini che identificano 115 nazioni estratti da un corpus testuale, rappresentati su un piano 2d.

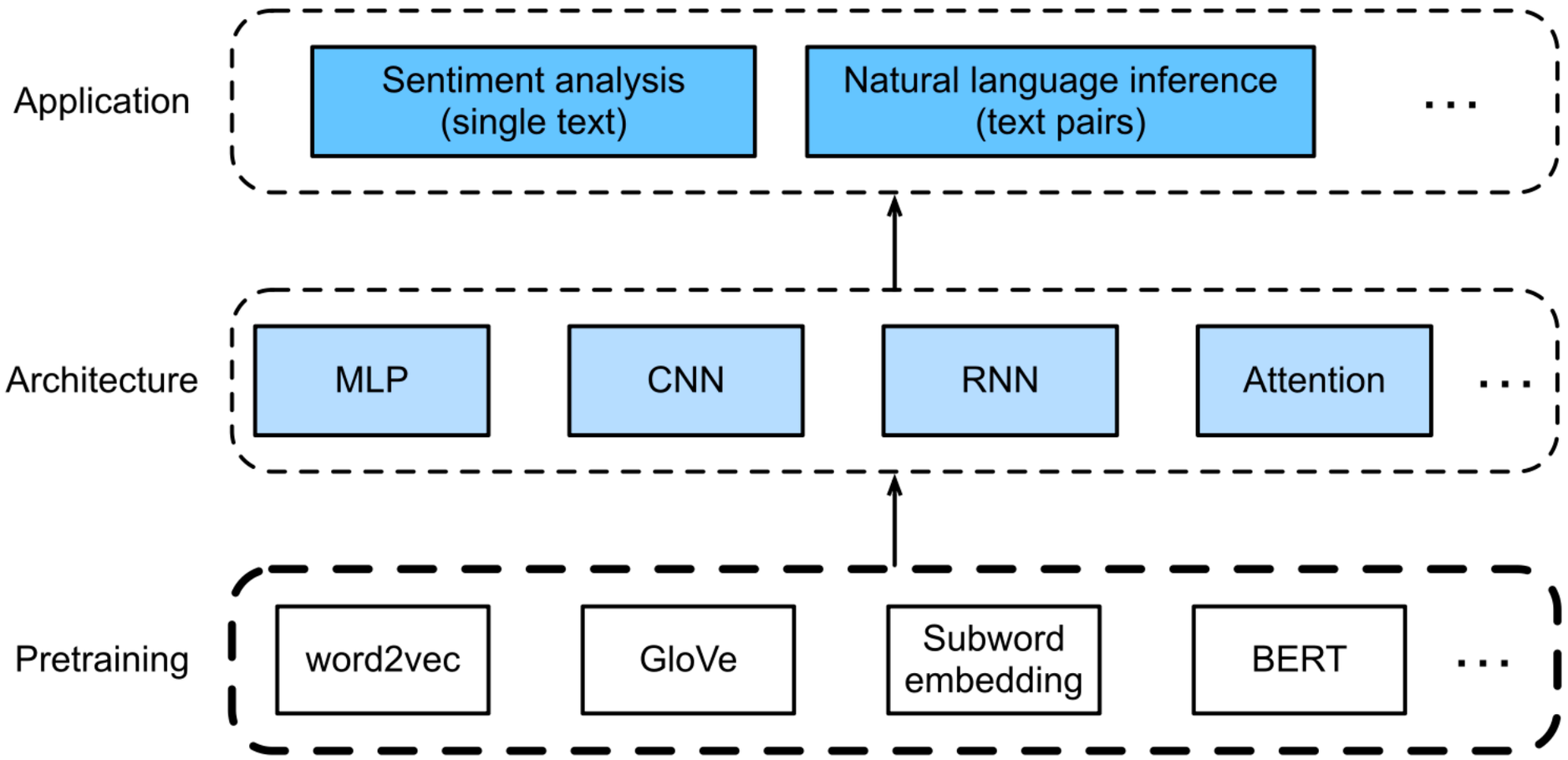


Esempio di operazioni su embeddings

- Essendo vettori, possiamo fare operazioni sugli embeddings, es:
  - *vector(“paris”)–vector(“france”)+vector("germany")*
- Impiegando il modello di embeddings *GloVe* addestrato sul testa estatto da Wikipedia otteniamo:
  - *berlin: 0.8015347*
  - *paris: 0.7623165*
  - *munich: 0.7013252*
  - *leipzig: 0.6616945*
  - *germany: 0.6540700*

DL e NLP

- Le rappresentazioni pretrained ottenute con approcci non supervisionati sono successivamente impiegate su architetture di DL in base al task da risolvere.



word2vec

- Il modello impiega 2 reti: **skip-gram** e **continuos bag of words** (CBOW).
- Il training è basato sulla stima delle probabilità condizionate di predire una certa parola in base a termini che occorrono nel suo intorno. Si segue sempre un approccio non supervisionato.

word2vec: skip-gram

- Assume che un termine può generare il testo circostante in una sequenza.
- Supponiamo di considerare la sequenza *“the”, “man”, “loves”, “his”, “son”*; e considerare il termine *loves* con parola centrale, e una finestra di 2 termini intorno al termine centrale.
- Il modello skip-gram valuta la seguente probabilità condizionata:

$$P(\text{”the”}, \text{”man”}, \text{”his”}, \text{”son”} \mid \text{”loves”})$$

- Se assumiamo che i termini siano generati in modo indipendente tra loro, possiamo riscriverla:

$$P(\text{”the”} \mid \text{”loves”}) \cdot P(\text{”man”} \mid \text{”loves”}) \cdot P(\text{”his”} \mid \text{”loves”}) \cdot P(\text{”son”} \mid \text{”loves”})$$

word2vec: skip-gram

- Ogni parola con indice ***i*** ha associati 2 vettori d-dimensionali, $v_i \in \mathbb{R}^d$ e $u_i \in \mathbb{R}^d$. Il primo impiegato quando la parola è usata centralmente, l'altro quando la parola appare nel contesto.
- La probabilità di generare un certo termine contestuale con indice **o** dato il termine centrale con indice ***c*** è definita mediante una operazione softmax nel seguente modo:

$$P(w_o \mid w_c) = \frac{\exp(\mathbf{u}_o^\top \mathbf{v}_c)}{\sum_{i \in \mathcal{V}} \exp(\mathbf{u}_i^\top \mathbf{v}_c)}$$

- Data una sequenza di testo lunga ***T***, dove $w^{(t)}$ indica la parola posizionata allo step ***t***, e assumendo che le parole contestuali siano generate in modo indipendente tra loro, per una finestra di lunghezza ***m***, la probabilità di generare tutti i termini contestuali è definita nel seguente modo:

$$\prod_{t=1}^{T} \prod_{-m \leq j \leq m,\, j \neq 0} P(w^{(t+j)} \mid w^{(t)})$$

word2vec: skip-gram (training)

- Ci poniamo l'obiettivo di massimizzare la probabilità, cioè minimizzare la funzione:

$$-\sum_{t=1}^{T} \sum_{-m \leq j \leq m,\ j \neq 0} \log P(w^{(t+j)} \mid w^{(t)})$$

- Se impieghiamo lo SGD, usiamo sequenze brevi per stimare il gradiente stocastico e aggiornare il modello. La stima è basta sul gradiente del logaritmo della probabilità condizionata data una coppia $\boldsymbol{w_o}$ e $\boldsymbol{w_c}$.

$$\log P(w_o \mid w_c) = \mathbf{u}_o^\top \mathbf{v}_c - \log \left(\sum_{i \in \mathcal{V}} \exp(\mathbf{u}_i^\top \mathbf{v}_c) \right)$$

- Una volta terminato l'apprendimento, i vettori $\boldsymbol{v_i}$ sono tipicamente impiegati come *embedding* associati ad un termine.

word2vec: Continuous Bag of Words (CBOW)

- Simile allo skip-gram ma assume che le parole contestuali generico la parola centrale.

$$P(\text{"loves"} \mid \text{"the"}, \text{"man"}, \text{"his"}, \text{"son"})$$

- Essendoci più parole contestuali, i vettori sono mediati. La probabilità condizionata di generare un termine w_c dati i termini contestuali $w_{o_1}, \cdots, w_{o_{2m}}$ è la seguente:

$$P(w_c \mid w_{o_1}, \ldots, w_{o_{2m}}) = \frac{\exp\left(\frac{1}{2m}\mathbf{u}_c^\top(\mathbf{v}_{o_1} + \ldots + \mathbf{v}_{o_{2m}})\right)}{\sum_{i \in \mathcal{V}} \exp\left(\frac{1}{2m}\mathbf{u}_i^\top(\mathbf{v}_{o_1} + \ldots + \mathbf{v}_{o_{2m}})\right)}$$

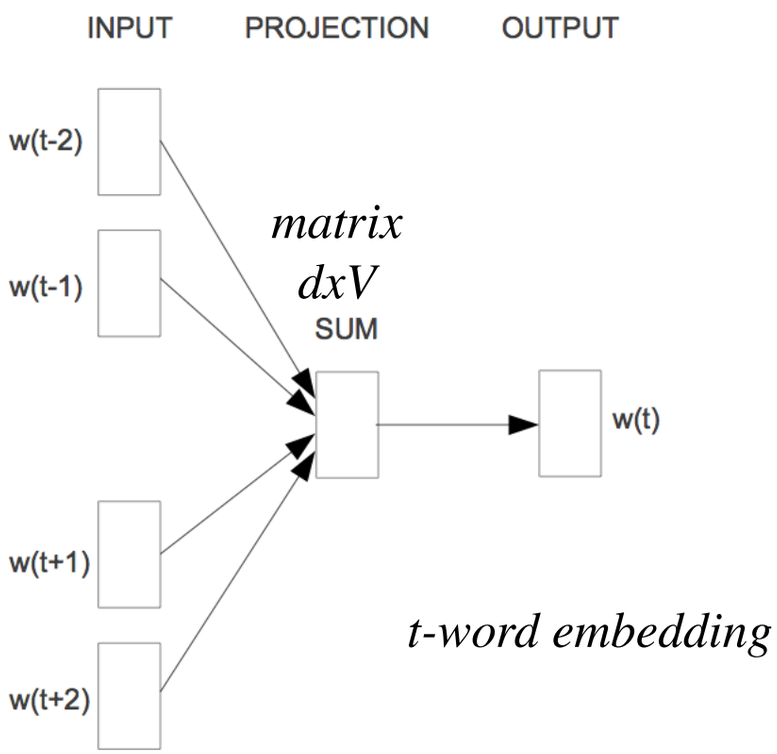
- Se consideriamo una sequenza di lunghezza T abbiamo:

$$\prod_{t=1}^{T} P(w^{(t)} \mid w^{(t-m)}, \ldots, w^{(t-1)}, w^{(t+1)}, \ldots, w^{(t+m)})$$

word2vec: recap

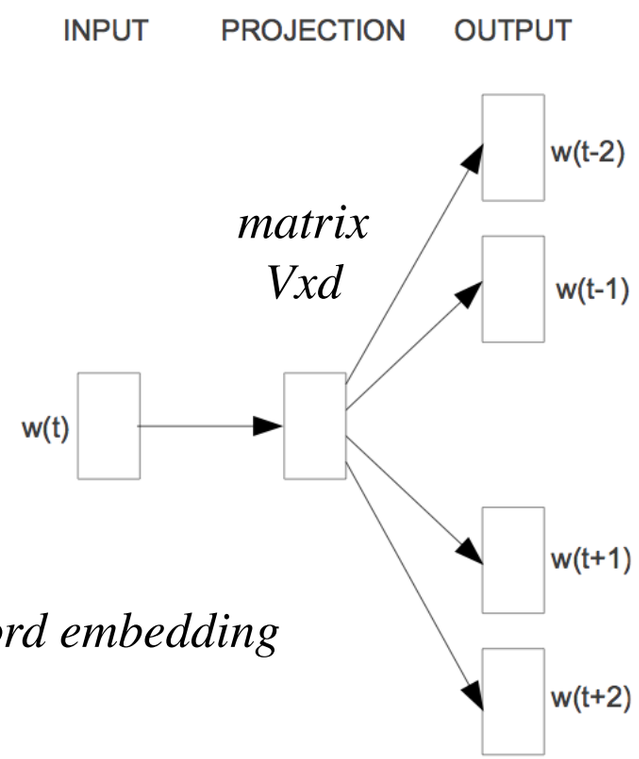
- Diagramma riassuntivo dei 2 modelli:

CBOW model



t-word embedding

Skip-gram model



t-word embedding

Word Embedding with Global Vectors (GloVe)

- Le co-occorrenze tra termini rappresentano informazioni importanti per costruire gli embeddings. Indichiamo con q_{ij} la probabilità condizionata $P(w_j|w_i)$ nel modello *skip-gram*:

$$q_{ij} = \frac{\exp(\mathbf{u}_j^\top \mathbf{v}_i)}{\sum_{k \in \mathcal{V}} \exp(\mathbf{u}_k^\top \mathbf{v}_i)}$$

- La parola $\mathbf{w_i}$ può presentarsi molte volte in un corpus. Tutte le parole contestuali che co-occorrono con $\mathbf{w_i}$ creano un multiset, dove $\mathbf{x_{ij}}$ indica il numero di volte che la parola $\mathbf{w_j}$ co-occorre con $\mathbf{w_i}$. La loss function è:

$$-\sum_{i \in \mathcal{V}} \sum_{j \in \mathcal{V}} x_{ij} \log\, q_{ij}$$

- Indichiamo con $\mathbf{x_i}$ il numero di parole contestuali dove compare wi come parola centrale, e avendo $p_{ij}=x_{ij}/x_i$, otteniamo:

$$-\sum_{i \in \mathcal{V}} x_i \sum_{j \in \mathcal{V}} p_{ij} \log\, q_{ij}$$

Word Embedding with Global Vectors (GloVe)

- La sommatoria interna è la cross-entropy tra la probabilità condizionata q_{ij} relativa alla predizione generata dal modello, e p_{ij} ottenuta analizzando le statistica dell'intero corpus.
- Per ridurre la complessità computazionale (soprattutto per generare q_{ij}) e per mitigare gli effetti generate dai termini che compaiono di rado nel corpus ma che possono assumere importanza elevata dalla cross entropy, il modello GloVe introduce alcune varianti.
- La nuova *loss function* è la seguente:

$$\sum_{i \in \mathcal{V}} \sum_{j \in \mathcal{V}} h(x_{ij}) \left(\mathbf{u}_j^\top \mathbf{v}_i + b_i + c_j - \log x_{ij} \right)^2.$$

- dove si introducono ad ogni parola sono associati 2 bias, $\boldsymbol{b_i}$ per le parole centrali e $\boldsymbol{c_i}$ per le parole impiegate nel contesto; il primo e ultimo termine nel termine a quadrato sono il termine di loss, e $h(x_{ij})$ genera un peso associato al termine di loss.

fastText model

- Ci sono relazioni morfologiche comuni tra molti vocaboli, es., tra *help* e *helps, helped, helping*; tra *dog* e *dogs* e tra *cat* e *cats*; tra *boy* e *boyfriend* e tra *girl* e *girlfriend*. Modelli come skip-gram ignorano queste relazioni, poiché ognuno di questi termini è rappresentato da un vettore distinto.
- Il modello **fastText** usa *subword embeddings*, dove ogni *subword* è un *n-gram* di caratteri. Ad ogni subword è associato un vettore.
  - Per esempio, la parola "where" genera le subwords “<wh”, “whe”, “her”, “ere”, “re>” impiegando una finestra di lunghezza 3.
- La rappresentazione di un termine $\mathbf{v_w}$ sarà la somma delle sue subwords $\mathbf{z_g}$:

$$\mathbf{v}_w = \sum_{g \in \mathcal{G}_w} \mathbf{z}_g.$$

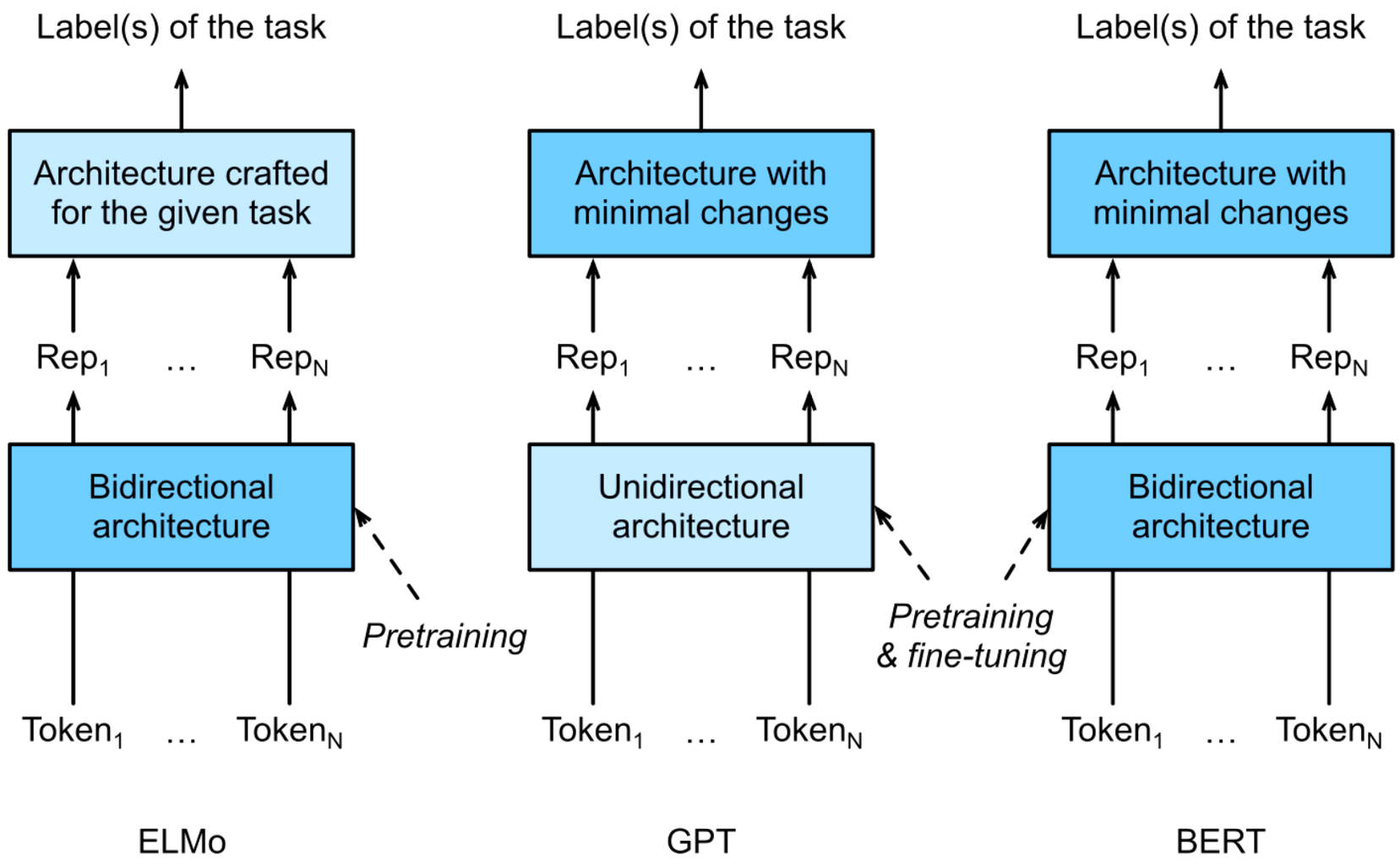
- Il resto del modello è basato su skip-gram.

Modelli context-indipendent e context-sensitive

- Nei modelli precedenti, data una parola, il vettore generato non dipende dal contesto attuale (approccio *context-independent*). Termini polisemici o relazioni semantiche del linguaggio naturale saranno perciò ignorate.
- Modelli come *ELMo* combinano le rappresentazioni intermedie ottenute da LSTM bidirezionali per ottenere una rappresentazione che dipende dalla sequenza in input (approccio *context-sensitive*).
  - La rappresentazione così ottenuta è solitamente combinata con quella ottenuta in modo context-independent (es. tramite GloVe) nei task successivi. Il modello impiegato da ELMo deve essere specifico per il task che si andrà ad affrontare, e perciò rimarrà costante.
- Per evitare di avere diversi modelli per ogni task, GPT pre-addestra un language model che sarà usato per rappresentare sequenze testuali. I parametri saranno poi *fine-tuned* in base all'ouput del task successivo. GPT è basato su Transformers. Il contesto analizzato da GPT sarà limitato alla parte antecedente al termine attuale, perciò non si analizza il contesto a destra del termine.

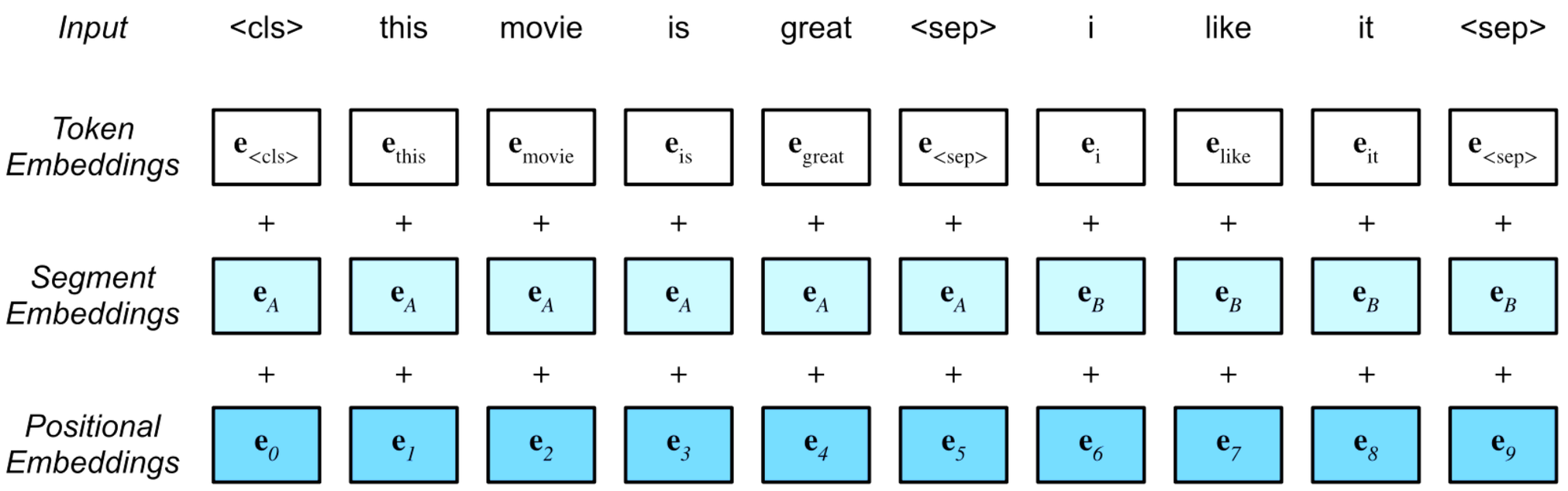
Bidirectional Encoder Representations from Transformers (BERT)

- BERT combina i due approcci appena descritti, rappresentando l'intero contesto mediante un approccio bidirezionale.
- È basato su Transformer encoders pre-addestrati. Un output layer specifico per il task da affrontare sarà di volta in volta addestrato da zero.



Bidirectional Encoder Representations from Transformers (BERT)

- L'input di BERT può essere una singolo testo o coppie di testi.
- Oltre agli *positional* embedding, si impiegano anche *segment* e *token* embeddings. Infatti, a differenza delle RNN, i Transformer richiedono tecniche specifiche per rappresentare internamente l'ordine relativo in cui i termini compaiono tra loro. Tali embedding sono ricavati durante la fase di training.



Bidirectional Encoder Representations from Transformers (BERT)

- Come pretraining (auxiliary) task si impiega il *Masked Language modeling*.
- Dato un corpus testuale, il 15% dei tokens saranno selezionati in modo random per il task di predizione. Al loro posto sarà presente un tag <mask>, es:
  - "this movie is great" becomes "this movie is <mask>"
- Un ulteriore auxiliary task specifico nello scenario in cui si hanno 2 testi in input è il *Next sentence prediction*. Dal corpus si estraggono coppie di frasi consecutive, e altrettante coppie di frasi che non sono consecutive. Il task è di classificazione binaria.

Bidirectional Encoder Representations from Transformers (BERT)

- BERT raggiunge prestazioni elevate su numero categorie di tasks, es.:
  - *single text classification* (e.g., sentiment analysis),
  - *text pair classification* (e.g., date due domande di Quora determinare se sono simili o no),
  - *question answering*,
  - *text tagging* (e.g., named entity recognition)