Imparare a predire la parola successiva

Quando facciamo speech recognition è impossibile individuare e distinguere perfettamente tutti i vari fonemi, ci sono più parole che possono corrispondere ad un determinato suono e anche l'essere umano a volta non riesce ad eliminare l'ambiguità e più che sentire lo deduce dal significato logico della frase.

Dobbiamo fare in modo che anche le macchine riescano a predire quale possa essere la parola successiva a quella che hanno appena elaborato, il che si può fare anche senza avere una comprensione cosciente (del significato semantico) della frase.

Metodo del Trigramma: consiste nell' andare a leggere in un enorme corpus (raccoglitore di frasi) e contare tutte le triplette di parole che contengono quella presa in esame.
Problemi:

- per predire una parola utilizzi solo le due precedenti? se dico "il mio" poi posso avere molte possibili terze parole.
- Con un insieme troppo alto rischio di trovare pochissimi match (nessun match non significa probabilità nulla) e sarò costretto ad ignorare le parole più lontane dalla successiva. Esempio: se ho "dinosauro pizza" non troverò mai match e dovrò prendere in esame solo pizza.
- Questo metodo non prende in considerazione la somiglianza semantica tra parole. Per esempio dopo "io mangio la" posso equivalentemente inserire "pizza, pasta,torta.."

Quello che vogliamo fare è convertire le parole in vettori di features sintattiche e semantiche. In questo modo possiamo anche estendere il contesto a più di 3 singole parole (il problema della difficoltà di matching non si pone perché si sfruttano le somiglianze logiche tra le parole, ovvero magari nel corpus non c'è "mangio la torta" ma se c'è "mangio la pasta" verrà comunque presa in considerazione perchè "torta" e "pasta" hanno un vettore descrittivo simile)

Prima di procedere con la spiegazione dell' architettura della rete neurale atta ad estrarre le features delle parole, introduciamo un caso risolubile con una stategia simile, ovvero:

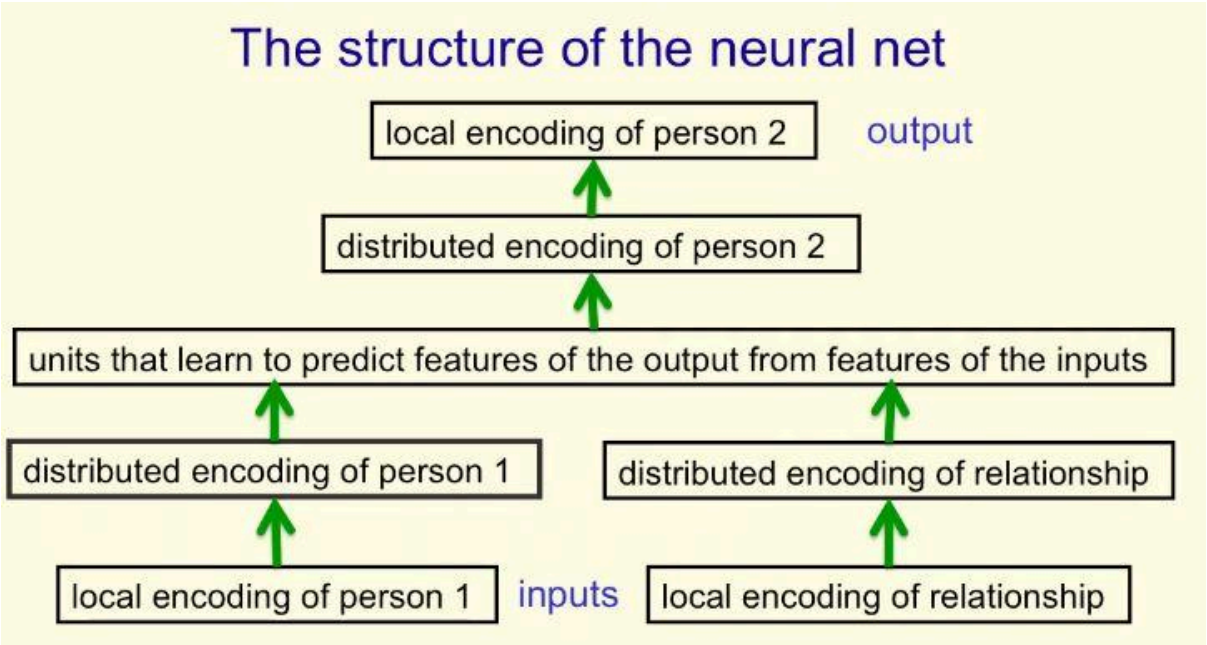
Albero genealogico mediante vettori di features:

Organizziamo la nostra conoscenza in triplette, ad esempio:
(Gohan, ha come padre, Goku)

Non vogliamo usare la logica del prim'ordine per fare deduzione, perché sarebbe è troppo lento il fare deduzione con essa, vogliamo quindi usare le reti neurali. L'idea è che a partire

da i primi due termini (il primo soggetto e la relazione) si riesca a trovare il terzo (ovvero il secondo soggetto).

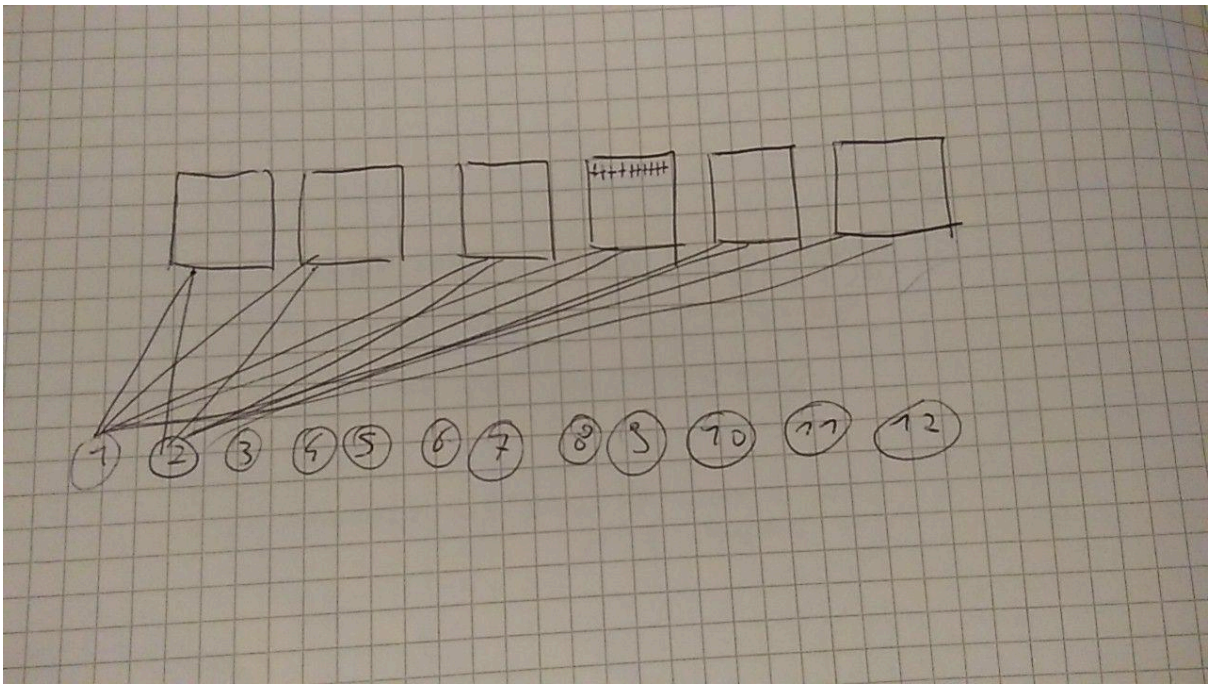
L’architettura di cui abbiamo bisogno è sommariamente questa:



Disponiamo quindi un neuroni di input per ogni singolo possibile primo elemento della tripletta (ovvero ogni nome proprio dei personaggi dell’ universo preso in esame) e parallelamente creiamo una seconda rete in cui gli input sono dedicati non più ai nomi dei personaggi ma alle relazioni (nel campo familiare sono circa 12.. "padre,madre, figlio,nipote..."). Questo per due motivi:

1) ci permette di incrementare linearmente la base di conoscenza
2) una rappresentazione più succinta potrebbe creare ambiguità nel capire precisamente quale neurone di input si è attivato
3) in questo modo, lo strato di input ci da solo ed esclusivamente l'informazione per capire di quale persona si parla, nulla di piu (non ci dara informazioni subliminali sulla possibile correlazione tra diverse persone, magari con una diversa rappresentazione, l’ input Gohan avrebbbe potuto triggerare anche il neuroni di Goku, anche se di un livello inferiore alla soglia)

Per spiegare meglio il passaggio tra “local encoding of p1” e “distributed encoding of p1” ho rappresentato la parte di albero interessata, in modo un po meno astratto:



Ciò che avviene a questo punto è che tutti i neuroni di input confluiscono in una sottorete comune che però è più piccola di quella dello strato di input. Per questo motivo gli input non verranno rirappresentati in modo unitario, ma solo come aventi o meno determinate caratteristiche. Queste features sono estrapolate e collezionate automaticamente dalla rete. Ogni neurone di questo strato intermedio corrisponderà ad una determinata features e in base al valore che verrà dato al peso del corrispettivo neurone di input capiremo se esso partecipa o meno a questa categoria. Per esempio un neurone hidden può triggerare molto neuroni appartenenti ad una generazione (tutti i genitori) e meno quelli di una generazione diversa (i figli). Oppure può triggerare solo quelli di una certa zona dell’ albero genealogico (tipo un neurone che da valore alto ai neuroni input di radish e bardack e goku perché appartenenti alla razza sayan, un valore medio a gohan e goten perche mezzosangue, ed un valore debole a chichi perché umana).

A questo punto è importante però che anche la rete delle relazioni sia addestrata e rappresenti la realtà secondo regole simili, ad esempio: un figlio mezzosangue ha un genitore di una razza e l’ altro genitore di un’altra razza.

Un po di numeri proveniente da un esempio pratico:
nomi: 24
gradi di parentela: 12
neuroni di features: 6
proposizioni nel dataset: 112

Tutte le sottoreti confluiranno in una rete intermediaria comunque che fara da punto di congiunzione e conterrà tutte le informazioni necessarie per fare l'inferenza finale.

Altre possibilità di utilizzo, invece di usare i primi due termini per predire il terzo attraverso le regole individuate dalla rete si possono utilizzare tutti e 3 i termini per individuare un fatto e usare le regole per stabilire se questo fatto è possibile o meno.
Es: (gohan, ha come padre junior) -> impossibile perché gohan è mezzsangue uomo/sayan mentre junior è un namecciano (bisogna avere una base di dati moooolto ampia per far funzionare tutto ciò).

Un altro strumento necessario per la comprensione di questo algoritmo è la **funzione di softmax**

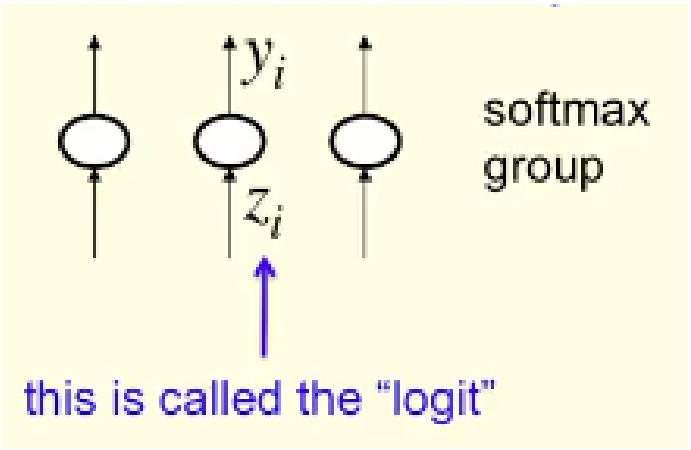
Softmax Function

L'algoritmo della distanza quadratica ha dei limiti.

- Se l'output desiderato è 1 mentre il risultato ottenuto è bassissimo, allora anche l'incremento dei pesi sarà tale.
- se vogliamo assegnare delle probabilità a delle classi di output che sono mutualmente esclusive, ci aspettiamo che la somma di tutte le percentuali assegnate ad un certo input sia 100%, ma la distanza quadratica media di questo non tiene conto.

Ci serve quindi una nuova funzione costo che rappresenti i risultati della rete sottoforma di probabilità, appunto la softmax.

Nel softmax i neuroni di output sono raccolti in gruppo.



La loro funzione di uscita (yi) non dipende solo dal loro ingresso e/o relativo bias ma tiene conto anche dell'input delle altre unità dello stesso gruppo.Nello specifico la formula è:

$$y_i = \frac{e^{z_i}}{\sum_{j \in group} e^{z_j}}$$

Quindi il valore di attivazione di ogni neurone di output sarà dato da: esponenziale del valore di ingresso del neurone, diviso la sommatoria dell' esponenziale degli ingressi dei neuroni dello stesso gruppo a cui appartiene il neurone.

Essendoci al denominatore una sommatoria di tutti gli e^z dello stesso gruppo, sappiamo che sommando tutte le yi otterremo 1 come totale e che tutte le yi avranno valore compreso tra 0 e 1. Quindi yi è una funzione che cambia al variare di zi

Softmax Equation:

$$\frac{\partial y_i}{\partial z_i} = y_i\,(1 - y_i)$$

Quanto cambia yi al variare di zi

Sostituendo la differenza dei quadrati con la softmax, dobbiamo modificare anche la funzione di costo utilizzando la:

Cross Entropy Cost Function: il negativo del logaritmo della probabilità di avere la giusta risposta.

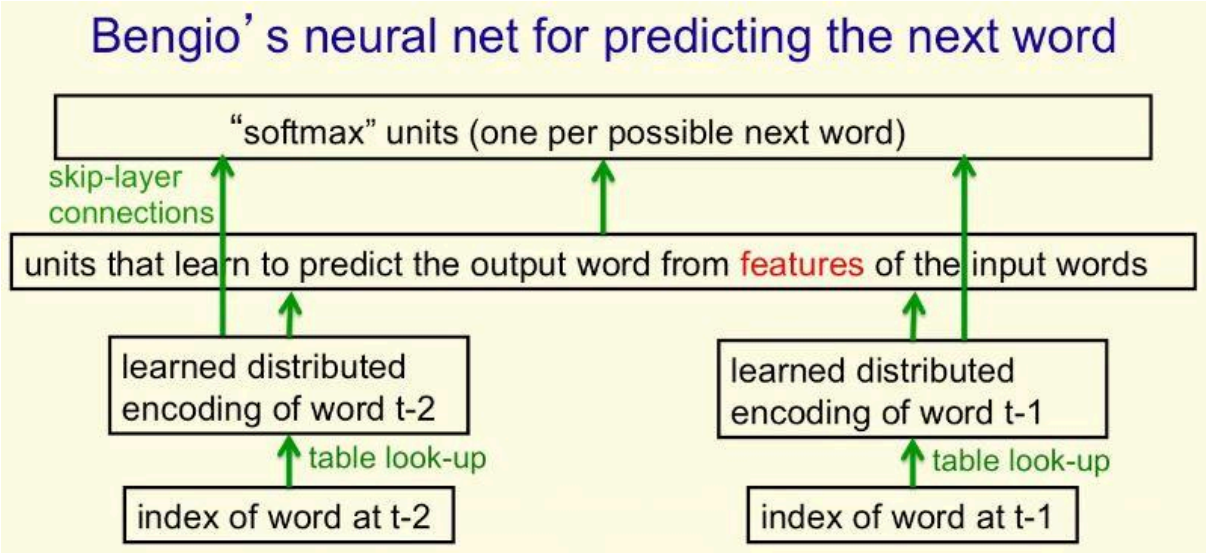
Ha un grande gradiente quando il target è 1 e l' output quasi nullo

$$C = -\sum_j t_j \log y_j$$

target value

Albero per l' estrazione automatica di features di una parola

La rete che costruiremo avrà un'architettura simile a questa:



I neuroni in fondo si occupano semplicemente di individuare una singola e specifica parola.Quindi ci serve un neuro per ogni parola che vogliamo usare per predire la successiva, per esempio nel caso dei trigrammi usavamo 2 parole per predire la terza.

Una volta individuata la parola, il neurone corrispondente confluirà in una piccola rete (learned distributed encoding of word t-2) che collega tutti i neuroni alle foglie dell' albero. Questa rete è più piccola di quella formata da tutti i neuroni foglia ed è tale perché non rappresenta tutte le parole ma solo le loro caratteristiche principali (features) ed è la rete stessa ad individuarle.

A questo punto uniamo attraverso una strato hidden tutti i gruppi di neuroni che rappresentano le features delle parole prese in esame. Questo strato hidden conterrà un dei neuroni che hanno imparato a predirre parole in base a quali features vengono triggerate.

Con l'algoritmo di "softmax" possiamo assegnare un grado di probabilità a queste parole candidati al triggeraemento dovuto alle features attivate dagli input.

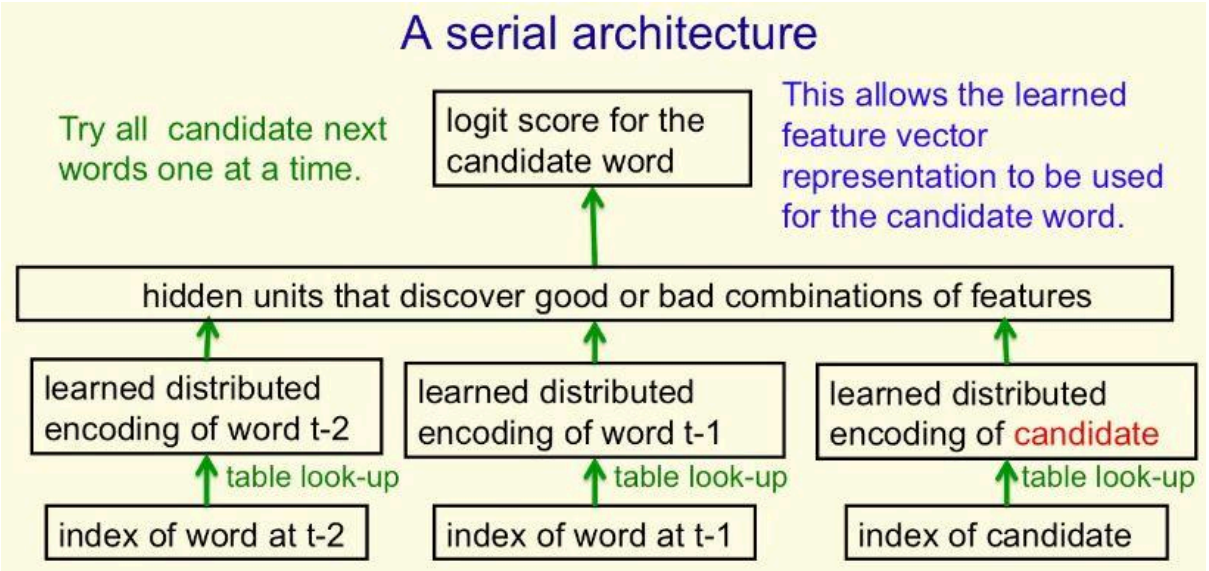
A volte si possono ottenere dei miglioramenti saltando dallo strato di "learned distibuted encoding" allo strato di output "softmax units".

Il problema di questo metodo è che lo strato di softmax deve gestire migliaia di possibili output visto che per esempio ogni verbo ha dei singolari, dei plurali, dei tempi, dei modi..

Se i neuroni di output hanno migliaia di collegamenti rischiano di dare over fitting se non si ha a disposizione un enorme quantitativo di dati. Se eliminiamo questi pesi invece abbiamo problemi a fare la predizione. Come possiamo risolvere questo problema?

Metodo N.1 "Aggiungendo il neurone di predizione"

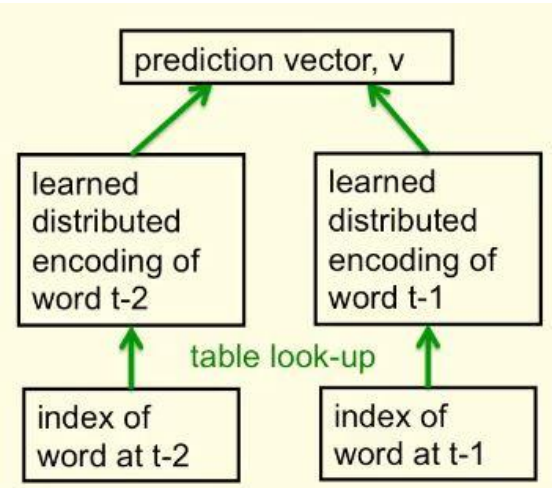
Dobbiamo innanzitutto modificare l'architettura della rete, aggiungendo ai neuroni di input un nuovo neurone che rappresenta la "parola candidata". L' output della rete sarà una probabilità che rappresenterà il grado di bontà di quella parola candidata.



Dovremmo ripercorrere la rete moltissime volte (ci saranno moltissime parole candidate) ma gran parte del lavoro va fatta solo la prima volta. Anche se è un parola futura, abbiamo a disposizione il suo encoding nel feature vector semplicemente perché avremmo già trovato quella parola in passato come non candidata ma come input.

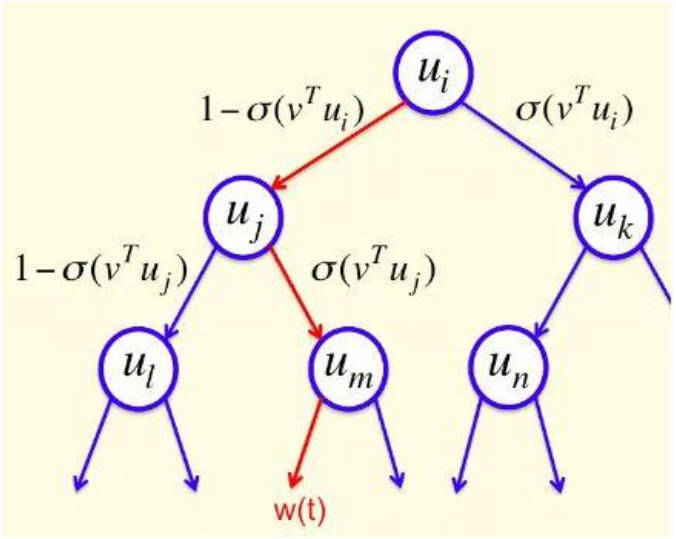
Metodo N.2 "predicendo il percorso sull' albero" → Scoperto da Minih e Hinton, 2009

- Si struttura un albero binario in cui le parole sono foglie.
- Si usano le altre parole per generare il vettore di predizione V con una banale rete neurale



- Poi si prendono tutte le parole candidate e se ne estrae il vettore di feature U

- applicare la funzione logistica al prodotto scalare di U e V per sapere su quale dei nodi figli spostarsi per scendere fino alle foglie che rappresenteranno la prossima parola



Vantaggi:

- Durante l' apprendimento dobbiamo lavorare solo su un percorso (perché sai già dove vai a finire) e non su tutto l' albero, quindi la complessita è log(N) invece che N
- Ad ogni nodo calcoliamo contemporaneamente la prob. di scendere a destra ed a sinistra.

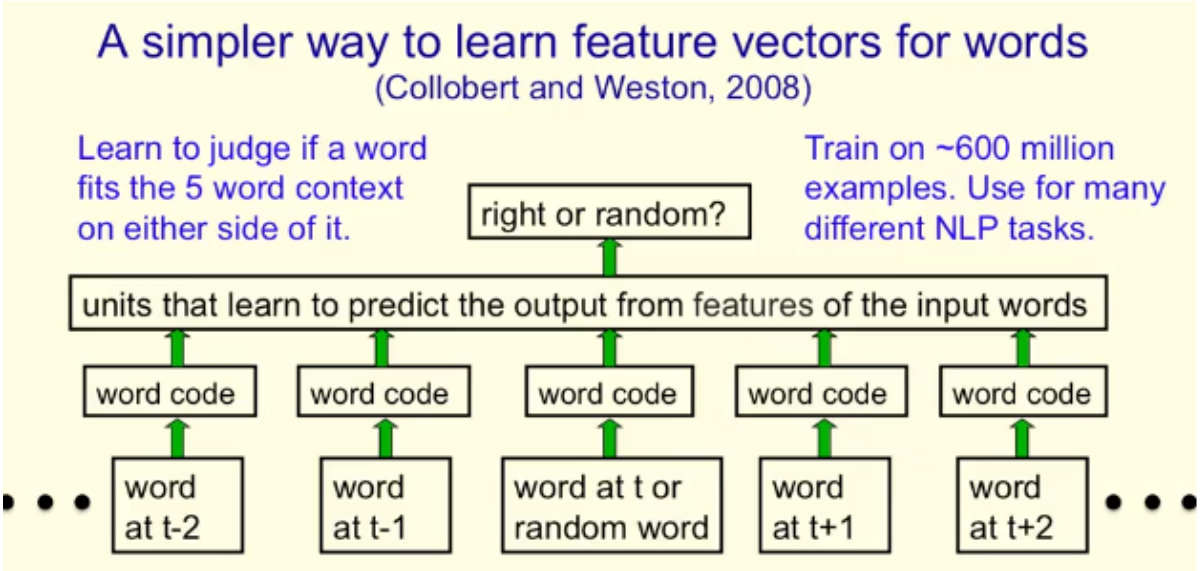
Svantaggi:

- In fase di utilizzo invece dei conoscere le probabilità di tuttel le possibili parole Ui.

Metodo N.3 "Utilizzando anche le parole successive" → Scoperto da Collobert e Weston, 2008

Metodo usato in vari ambiti del Natural Language Processing. Il loro obiettivo non era quello di predirre la parola successiva ma quello di produrre un ottimo vettore di features per le parole. Per far ciò hanno deciso di utilizzare parole precedenti e successive per formare il contesto.

Al centro posizionavano o la parola corretta o una casuale e la rete produceva un valore alto quando la parola era corretta ed uno basso quando era quella casuale.



Intuizione di un vettore di features
Un buon modo per stabilire se i vettori di feature appresi sono corretti è quello di stamparli su uno spazio bidimensionale (spazio cartesiano). E noteremo che vettori simili saranno fisicamente vicini e scopriremo quali parole la rete reputa correlati.

Strumento consigliato e da approfondire: **t-sne**

Semplicemente sapendo che quelle parole compaiono sempre insieme, la rete apprende che ci sono delle correlazioni anche a livello semantico, senza conoscere il significato e senza bisogno di supervisionamento

approfondimento 1:
https://www.coursera.org/learn/neural-networks/discussions/weeks/5/threads/yt4OUL1rEee0-w5wvP3L9A

approfondimento 2:
https://www.coursera.org/learn/neural-networks/discussions/weeks/5/threads/aCxte8N5EeaGMhL6_lg_XA

Part of a 2-D map of the 2500 most common words