



Università
degli Studi
di Palermo



Semantica Vettoriale ed Embeddings

CORSO DI NATURAL LANGUAGE PROCESSING (ELABORAZIONE DEL LINGUAGGIO NATURALE)

a.a. 2025/2026

Prof. Roberto Pirrone



Semantica lessicale

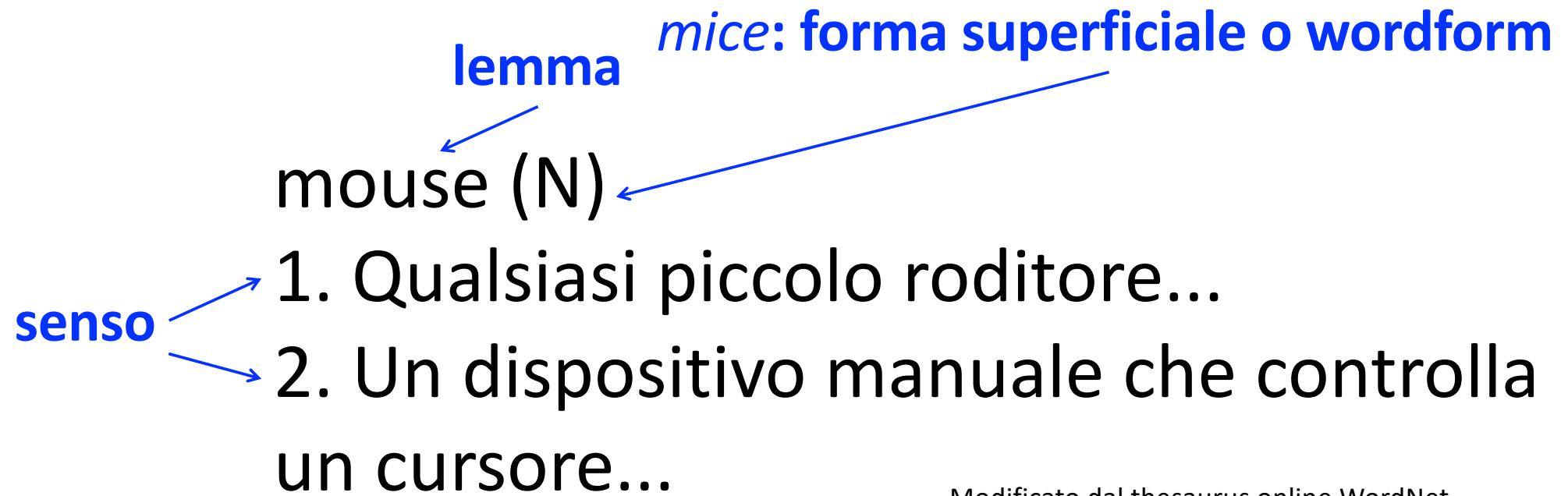
Cosa significano le parole?

- I metodi basati su N-gram o sulla classificazione di testi che abbiamo visto finora trattano le parole semplicemente come stringhe (o indici in una lista di vocabolario)
- *Questo approccio non è molto soddisfacente!*
- Nei corsi introduttivi di logica, ad esempio, il significato di “dog” è rappresentato come **DOG**, e quello di “cat” come **CAT**
→ $\forall x \text{ DOG}(x) \rightarrow \text{MAMMAL}(x)$
- Una vecchia battuta linguistica di **Barbara Partee (1967)**:
 - Q: What's the meaning of life?
 - A: LIFE

Cosa dovrebbe fare una teoria del significato delle parole

- Cosa dovrebbe offrirci una teoria del significato lessicale?
- Vediamo alcuni desiderata tratti dalla ***semantica lessicale***, lo studio linguistico del significato delle parole.

Lemmi e sensi



Modificato dal thesaurus online WordNet

Un **senso** (o “concetto”) è la **componente di significato** di una parola. I lemmi possono essere **polisemici**, cioè avere più sensi.

Relazioni tra sensi: Sinonimia

- I ***sinonimi*** hanno lo stesso significato in alcuni o in tutti i contesti.
 - filbert / hazelnut
 - couch / sofa
 - big / large
 - automobile / car
 - vomit / throw up
 - water / H₂O

Relazioni tra sensi: Sinonimia

- In realtà, **non esistono sinonimi perfetti.**
- Anche se molti aspetti del significato coincidono, possono comunque differire per:
 - cortesia,
 - registro linguistico,
 - slang,
 - genere, ecc.

Relazioni tra sensi: Sinonimia

water/H₂O

dire “H₂O” in una guida di surf?

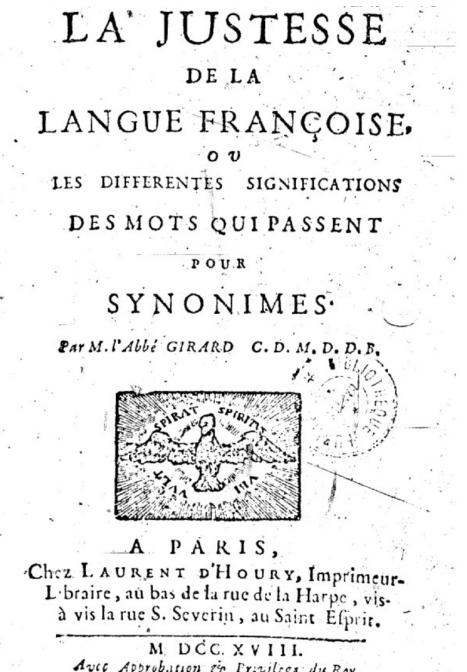
big/large

my big sister != my large sister

Il principio linguistico di contrasto

- *Differenza nella forma → differenza nel significato*
- Formulato da Abbé Gabriel Girard (1718):
«Non credo che esista una parola perfettamente sinonima in nessuna lingua.»

je ne crois pas qu'il y ait de mot synonyme dans aucune Langue. Je le dis par con-



Relation: Somiglianza

Le parole con significati simili non sono sinonimi, ma condividono
alcuni elementi semantici.

car, bicycle

cow, horse

Chiediamo agli umani quanto sono simili due parole

word1	word2	similarity
vanish	disappear	9.8
behave	obey	7.3
belief	impression	5.95
muscle	bone	3.65
modest	flexible	0.98
hole	agreement	0.3

Relazione: Associazione tra parole

- Detta anche **word association**.
- Le parole possono essere correlate in qualsiasi modo, magari attraverso un frame o un campo semantico.
 - coffee, tea: *simili*
 - coffee, cup: *correlate*, non simili

Campo semantico

- Le parole che ***appartengono a uno stesso dominio semantico*** intrattengono relazioni strutturate fra loro.

hospitals

surgeon, scalpel, nurse, anaesthetic, hospital

restaurants

waiter, menu, plate, food, menu, chef

houses

door, roof, kitchen, family, bed

Campo semantico

- Topic modeling
 - metodo di **apprendimento non supervisionato** dei campi semantici.
 - Fa uso della ***Latent Dirichlet Allocation (LDA)***
 - Modello generativo: un documento è generato da una **combinazione di argomenti (topics)**.
 - Ogni **topic** ha una **distribuzione di probabilità** sulle parole.
 - Le **parole** sono le **uniche variabili osservabili**.
 - La **distribuzione di Dirichlet** è usata come **prior** per i topics e per le parole.

Relazione: Antonimia

- Gli **antonimi** sono sensi opposti rispetto a una **singola dimensione di significato**.
- Per il resto, possono essere molto simili!

dark/light	short/long	fast/slow	rise/fall
hot/cold	up/down		in/out
- Formalmente, gli antonimi possono:
 - definire un'**opposizione binaria**, oppure trovarsi ai **poli opposti di una scala**
 - long/short, fast/slow
 - essere **reversivi**
 - rise/fall, up/down

Wordnet e Framenet

Lessico dei sensi: WordNet

- Oltre alla **sinonimia** e all'**antonimia**, esistono molte altre relazioni tra sensi:

Relazioni tassonomiche:

- **Iponimia** (relazione di sotto-classe) → *car* → *vehicle*
- **Iperonimia** (relazione di sovra-classe) → *mammal* → *dog*
- Spesso rappresentate come una **gerarchia IS-A**.

Lessico dei sensi: WordNet

- Oltre alla **sinonimia** e all'**antonimia**, esistono molte altre relazioni tra sensi:
- **Meronimia** (*relazione parte-tutto*):
- *car* → *wheel*
 - *wheel* è il **meronimo** di *car*
 - *car* è l'**olonimo** di *wheel*

Lessico dei sensi: WordNet

- Oltre alla **sinonimia** e all'**antonomia**, esistono molte altre relazioni tra sensi:
- **Polisemia strutturata**: relazione semantica tra *sensi diversi della stessa parola*.
Esempi: *bank, university, hospital*
 - **BUILDING ↔ ORGANIZATION**

Lessico dei sensi: WordNet

- Oltre alla **sinonimia** e all'**antonomia**, esistono molte altre relazioni tra sensi:
- **Metonimia**: relazione semantica tra **un aspetto di un'entità e altri aspetti o l'intera entità**.
 - *The White House* → indica l'**Amministrazione americana**
 - *I love Jane Austen* → indica le **opere** dell'autrice

Lessico dei sensi: WordNet

- WordNet è un **database lessicale** che rappresenta le relazioni tra parole in molte lingue.
- Le sue voci hanno una struttura simile a quella di un dizionario.
- Esempio: *chump*

Gloss: a person who is gullible and easy to take advantage of.

Synset: {*chump*¹, *fool*², *gull*¹, *mark*⁹, *patsy*¹, *fall guy*¹,
*sucker*¹, *soft touch*¹, *mug*²}

Lessico dei sensi: WordNet

- Supersensi
 - raggruppano i synset in categorie semantiche più ampie, per semplificare l'analisi linguistica e semantica.

Category	Example	Category	Example	Category	Example
ACT	<i>service</i>	GROUP	<i>place</i>	PLANT	<i>tree</i>
ANIMAL	<i>dog</i>	LOCATION	<i>area</i>	POSSESSION	<i>price</i>
ARTIFACT	<i>car</i>	MOTIVE	<i>reason</i>	PROCESS	<i>process</i>
ATTRIBUTE	<i>quality</i>	NATURAL EVENT	<i>experience</i>	QUANTITY	<i>amount</i>
BODY	<i>hair</i>	NATURAL OBJECT	<i>flower</i>	RELATION	<i>portion</i>
COGNITION	<i>way</i>	OTHER	<i>stuff</i>	SHAPE	<i>square</i>
COMMUNICATION	<i>review</i>	PERSON	<i>people</i>	STATE	<i>pain</i>
FEELING	<i>discomfort</i>	PHENOMENON	<i>result</i>	SUBSTANCE	<i>oil</i>
FOOD	<i>food</i>			TIME	<i>day</i>

Lessico dei sensi: WordNet

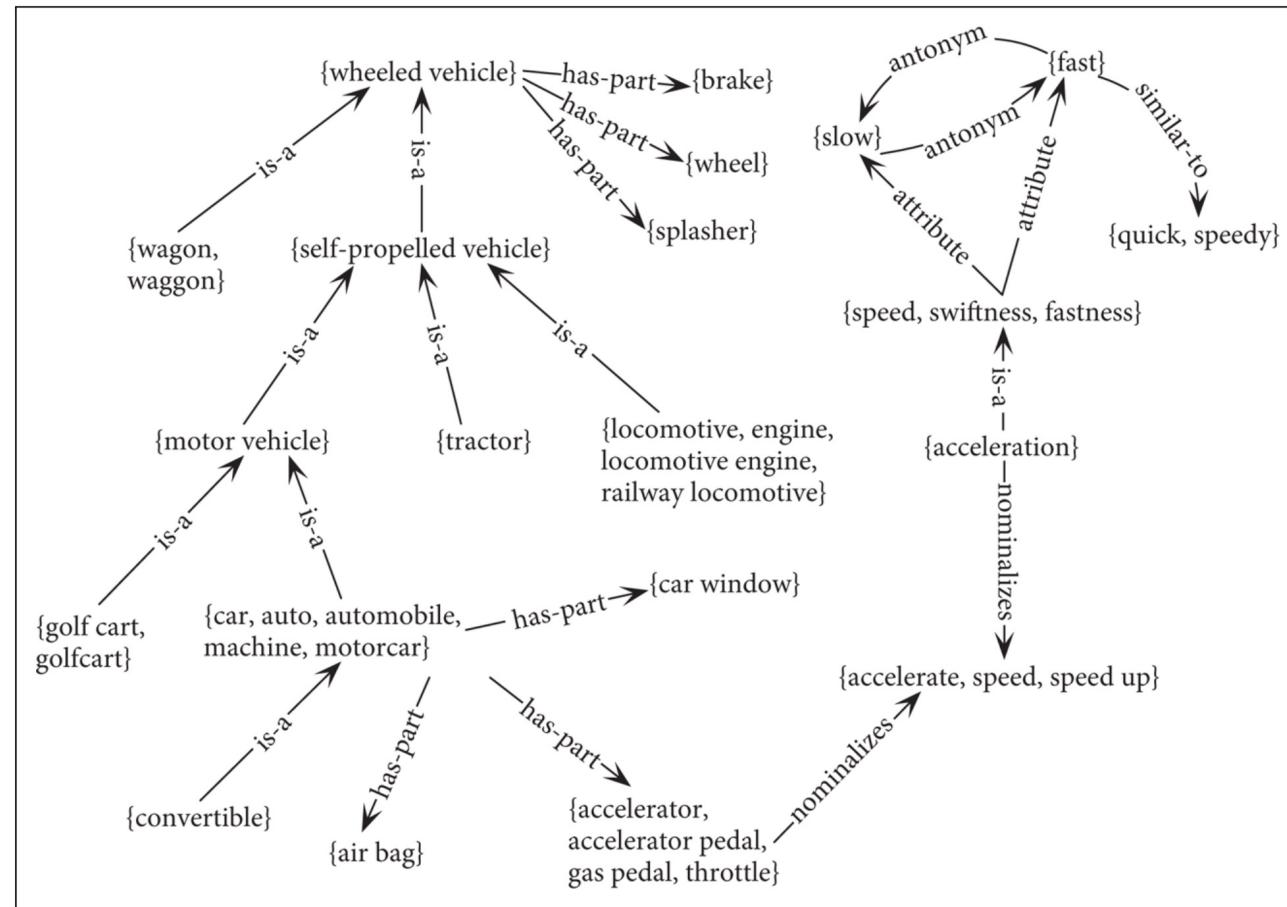
- Relazioni

Relation	Also Called	Definition	Example
Hypernym	Superordinate	From concepts to superordinates	<i>breakfast</i> ¹ → <i>meal</i> ¹
Hyponym	Subordinate	From concepts to subtypes	<i>meal</i> ¹ → <i>lunch</i> ¹
Instance Hypernym	Instance	From instances to their concepts	<i>Austen</i> ¹ → <i>author</i> ¹
Instance Hyponym	Has-Instance	From concepts to their instances	<i>composer</i> ¹ → <i>Bach</i> ¹
Part Meronym	Has-Part	From wholes to parts	<i>table</i> ² → <i>leg</i> ³
Part Holonym	Part-Of	From parts to wholes	<i>course</i> ⁷ → <i>meal</i> ¹
Antonym		Semantic opposition between lemmas	<i>leader</i> ¹ ⇔ <i>follower</i> ¹
Derivation		Lemmas w/same morphological root	<i>destruction</i> ¹ ⇔ <i>destroy</i> ¹

Relation	Definition	Example
Hypernym	From events to superordinate events	<i>fly</i> ⁹ → <i>travel</i> ⁵
Troponym	From events to subordinate event	<i>walk</i> ¹ → <i>stroll</i> ¹
Entails	From verbs (events) to the verbs (events) they entail	<i>snore</i> ¹ → <i>sleep</i> ¹
Antonym	Semantic opposition between lemmas	<i>increase</i> ¹ ⇔ <i>decrease</i> ¹

Word senses thesaurus: WordNet

- Un frammento del grafo semantico tra le relazioni di WordNet



Semantica dei frame e FrameNet

- Le relazioni tra word senses non sono sufficienti a cogliere l'intera semantica di una frase.
- *reservation, flight, travel, buy, price, cost, fare, rates, meal, plane*
- Oltre a sinonimia, iperonimia e simili, esiste una semantica comune nel contesto dei *viaggi in aereo*.

Semantica dei frame e FrameNet

- Frame semantico

- Un insieme di parole che denotano **prospettive** o **partecipanti** in un determinato tipo di evento.
- **Evento:** *commercial transaction*
- **Verbi:** *buy, sell, pay* (tre prospettive)
- Ruoli semanticici: buyer, seller, goods, money... (partecipanti)

Semantica dei frame e FrameNet

- Semantic Role labeling

Sam bought the book from Ling

Semantica dei frame e FrameNet

- Semantic Role labeling

*Sam **bought** the book from Ling*

Perspective: buy

Semantica dei frame e FrameNet

- Semantic Role labeling

Sam bought the book from Ling

buyer

Semantica dei frame e FrameNet

- Semantic Role labeling

Sam bought the book from Ling

goods

Semantica dei frame e FrameNet

- Semantic Role labeling

Sam bought the book from Ling

seller

Semantica dei frame e FrameNet

- [FrameNet](#) è un progetto esteso per il semantic role labeling, che codifica migliaia di frame in molte lingue.
 - **Core roles:** ruoli semantici specifici del *frame*
 - **Non core roles:** ruoli generici, come quelli relativi a tempo, posizione, ecc.
 - **Relazioni** tra i frame

Semantica dei frame e FrameNet

- Frame *change_of_position_on_a_scale*
 - **Definizione:**
Questo *frame* è costituito da parole che indicano il **cambiamento della posizione di un elemento (Item)** su una **scala (Attribute)**, da un **punto di partenza (Initial value)** a un **punto di arrivo (Final value)**.

Semantica dei frame e FrameNet

- Frame change of position on a scale

Core Roles	
ATTRIBUTE	The ATTRIBUTE is a scalar property that the ITEM possesses.
DIFFERENCE	The distance by which an ITEM changes its position on the scale.
FINAL_STATE	A description that presents the ITEM's state after the change in the ATTRIBUTE's value as an independent predication.
FINAL_VALUE	The position on the scale where the ITEM ends up.
INITIAL_STATE	A description that presents the ITEM's state before the change in the ATTRIBUTE's value as an independent predication.
INITIAL_VALUE	The initial position on the scale from which the ITEM moves away.
ITEM	The entity that has a position on the scale.
VALUE_RANGE	A portion of the scale, typically identified by its end points, along which the values of the ATTRIBUTE fluctuate.
Some Non-Core Roles	
DURATION	The length of time over which the change takes place.
SPEED	The rate of change of the VALUE.
GROUP	The GROUP in which an ITEM changes the value of an ATTRIBUTE in a specified way.
VERBS: dwindle advance edge climb explode decline fall decrease fluctuate diminish gain dip grow double increase drop jump	

Semantica dei frame e FrameNet

- Frame *change_of_position_on_a_scale*
- *a steady increase* [INITIAL VALUE *from 9.5*] [FINAL VALUE *to 14.3*] [*ITEM in dividends*]
- Frame *cause_change_of_position_on_a_scale*
- [AGENT They] *raised* [ITEM the price of their soda] [DIFFERENCE by 2%]

Rappresentazione del sentimento

Connotazioni (sentiment)

- Le parole hanno **significati affettivi**
 - Connotazioni positive (*happy*)
 - Connotazioni negative (*sad*)
- Le connotazioni possono essere **sottili**:
 - Connotazione positiva: *copy, replica, reproduction*
 - Connotazione negativa: *fake, knockoff, forgery*

Valutazione (sentiment!)

- Valutazione positiva: *great, love*
- Valutazione negativa: *terrible, hate*

Tipologia degli stati affettivi secondo Scherer

- **Emotion:** stato breve, organicamente sincronizzato... valutazione di un evento importante
angry, sad, joyful, fearful, ashamed, proud, elated
- **Mood:** cambiamento diffuso, non causato, a bassa intensità e di lunga durata nel sentimento soggettivo
cheerful, gloomy, irritable, listless, depressed, buoyant
- **Atteggiamenti interpersonali:** atteggiamento affettivo verso un'altra persona in una specifica interazione
friendly, flirtatious, distant, cold, warm, supportive, contemptuous
- **Attitudini:** convinzioni e disposizioni durature, colorate affettivamente, nei confronti di oggetti o persone
liking, loving, hating, valuing, desiring
- **Tratti personali:** disposizioni di personalità stabili e tendenze comportamentali tipiche
nervous, anxious, reckless, morose, hostile, jealous

Tipologia degli stati affettivi secondo Scherer

- **Emotion:** stato breve, organicamente sincronizzato... valutazione di un evento importante
angry, sad, joyful, fearful, ashamed, proud, elated
- **Mood:** cambiamento diffuso, non causato, a bassa intensità e di lunga durata nel sentimento soggettivo
cheerful, gloomy, irritable, listless, depressed, buoyant
- **Atteggiamenti interpersonali:** atteggiamento affettivo verso un'altra persona in una specifica interazione
friendly, flirtatious, distant, cold, warm, supportive, contemptuous
- **Attitudini:** convinzioni e disposizioni durature, colorate affettivamente, nei confronti di oggetti o persone
liking, loving, hating, valuing, desiring
- **Tratti personali:** disposizioni di personalità stabili e tendenze comportamentali tipiche
nervous, anxious, reckless, morose, hostile, jealous

Due famiglie di teorie dell'emozione

- Emozioni atomiche di base
 - Un elenco finito di 6 o 8 emozioni, dalle quali vengono generate tutte le altre.
- Dimensioni delle emozioni
 - **Valence** (positiva / negativa)
 - **Arousal** (forte / debole)
 - **Control**

Le sei emozioni di base di Ekman

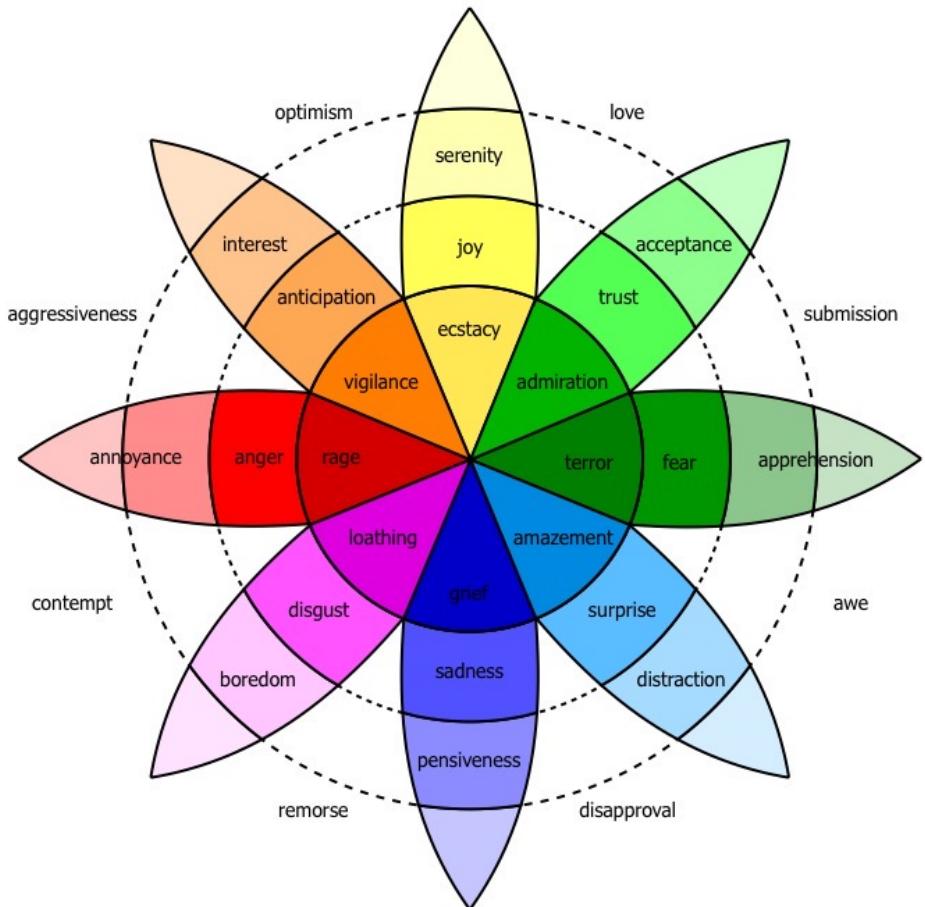
Surprise, happiness, anger, fear, disgust, sadness

Ekman &
Matsumoto
1989



La ruota delle emozioni di Plutchick

- 8 emozioni di base in quattro coppie opposte:
 - joy–sadness
 - anger–fear
 - trust–disgust
 - anticipation–surprise



Dimensioni delle emozioni

- Le tre dimensioni affettive sono usate per misurare le parole

	Word	Score		Word	Score
Valence	love	1.000		toxic	0.008
	happy	1.000		nightmare	0.005
Arousal	elated	0.960		mellow	0.069
	frenzy	0.965		napping	0.046
Dominance	powerful	0.991		weak	0.045
	leadership	0.983		empty	0.081

Valori dal Lessico NRC VAD Lexicon (Mohammad 2018)

Semantica vettoriale

Modelli computazionali del significato delle parole

- Possiamo costruire una teoria su come **rappresentare il significato delle parole** che soddisfi almeno alcuni dei nostri desiderata?
- Introduciamo la ***semantica vettoriale***
- Il modello standard nell'elaborazione del linguaggio
- Raggiunge molti dei nostri obiettivi

Vector semantics: idea 1

- Wittgenstein, PI #43:
"Il significato di una parola è il suo uso nel linguaggio"

Vector semantics: idea 1

- Definiamo "usage":
le parole sono definite dal loro “ambiente” (le parole attorno a loro)
- Zellig Harris (1954):
«Se **A** e **B** hanno ambienti quasi identici, diciamo che sono **sinonimi.**»

Vector semantics: idea 1

- Supponiamo di leggere queste frasi:
 - *Ong choi is delicious sautéed with garlic.*
 - *Ong choi is superb over rice.*
 - *Ong choi leaves with salty sauces.*
- E di avere già visto queste:
 - ...*spinach sautéed with garlic over rice*
 - *Chard stems and leaves are delicious*
 - *Collard greens and other salty leafy greens*
- Conclusioni:
 - *Ong choi è una **verdura a foglia verde**, simile a *spinach*, *chard* o *collard greens*.*
 - Possiamo giungere a questa conclusione grazie a parole come “*leaves*”, “*delicious*” e “*sautéed*”.

Idea 2: Il significato come un punto in uno spazio (Osgood et al. 1957)

- Ad esempio le tre dimensioni affettive di una parola
 - **valence**: piacevolezza
 - **arousal**: intensità dell'emozione
 - **dominance**: grado di controllo esercitato
- Da qui la rappresentazione come un vettore (in questo caso tridimensionale)

Definire il significato come un punto nello spazio in base alla distribuzione linguistica.

- Ogni parola → un vettore (non semplicemente "good" o " w_{45} ")
- Parole simili sono ***vicine nello spazio semantico***
- Costruiamo questo spazio ***in modo automatico***, osservando ***quali parole compaiono vicine nel testo***.



Definiamo il significato di una parola come un vettore

- Chiamato **embedding** perché è *inserito (embedded)* all'interno di uno spazio
- È il **metodo standard per rappresentare il significato** in NLP.

👉 Ogni moderno algoritmo di NLP utilizza gli **embeddings** come rappresentazione del significato delle parole.

Utile per analizzare le **similarità** tra parole.

Perché vettori?

- Consideriamo la sentiment analysis:
 - Usando direttamente le **parole**, una *feature* è semplicemente l'identità della parola.
 - Esempio - Feature 5: The previous word was "terrible"
 - È necessario che *esattamente la stessa parola* compaia nel training set e nel test set
 - Con gli embeddings:
 - La feature è un word vector
 - Ad esempio: la parola precedente era il vettore [35,22,17...]
 - Potremmo trovare una parola simile nel test set: [34,21,14]
 - Quindi possiamo **generalizzare a parole simili** non osservate prima!!!

Embedding TF-IDF

Affronteremo due tipi di embeddings

- tf-idf

- Tipico dell'Information Retrieval
- Baseline comune per molti approcci
- Vettori **sparsi**
- Le parole sono rappresentate da (**una semplice funzione dei**) conteggi delle **parole vicine** nel testo.

- Word2vec

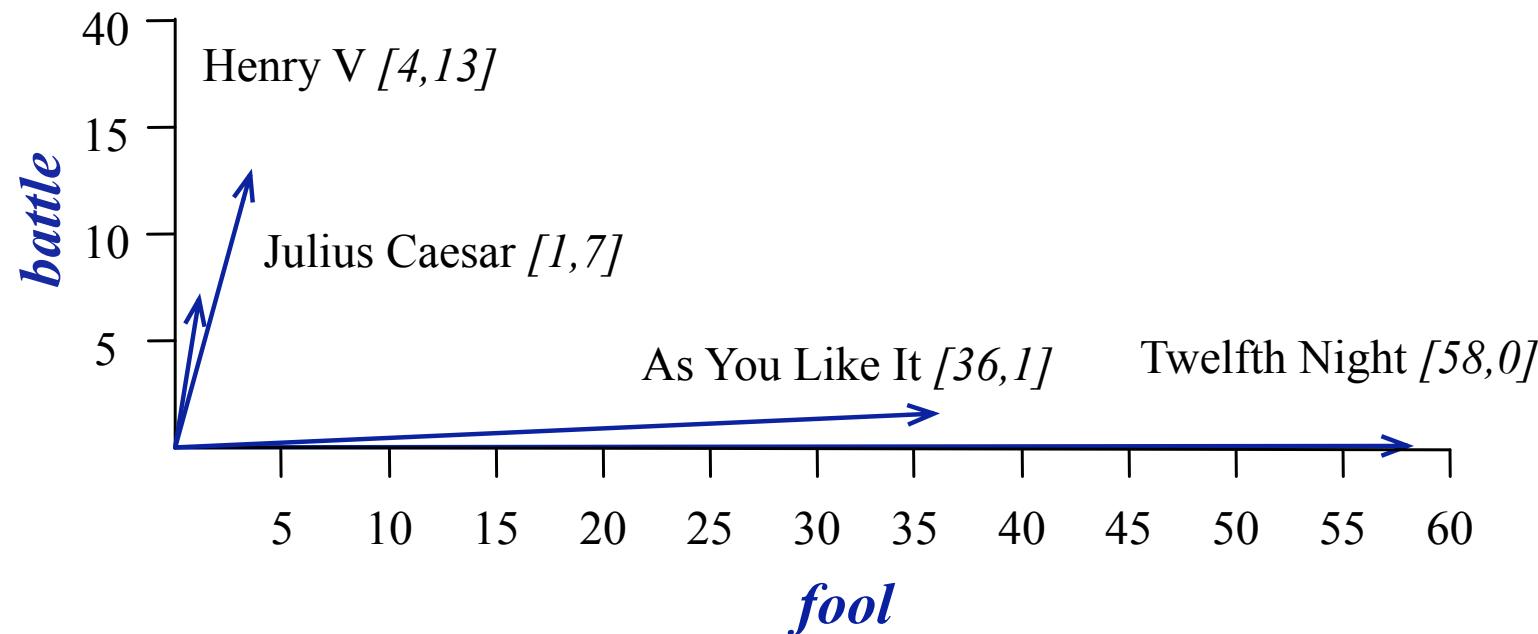
- Vettori **densi**
- La rappresentazione viene creata addestrando un classificatore a **prevedere se una parola è probabile che compaia nelle vicinanze**.
- Esiste un'estensione chiamata **contextual embeddings**

Term-document matrix

Ogni documento è rappresentato come un vettore di parole

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	1	0	7	13
good	114	80	62	89
fool	36	58	1	4
wit	20	15	2	3

Visualizzazione dei vettori dei documenti



I vettori sono la base dell'Information Retrieval

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	1	0	7	13
good	114	80	62	89
fool	36	58	1	4
wit	20	15	2	3

I vettori sono simili per le due commedie

Ma le commedie sono differenti dagli altri due documenti

Hanno più *fools* e *wit* e meno *battles*.

Ma anche le parole possono essere vettori!

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	1	0	7	13
good	114	80	62	89
fool	36	58	1	4
wit	20	15	2	3

battle è “quel tipo di parola che occorre in Giulio Cesare ed Enrico V”

fool è “quel tipo di parola che occorre nelle commedie, specialmente La Dodicesima Notte”

Più comunemente: word-word matrix (o "term-context matrix")

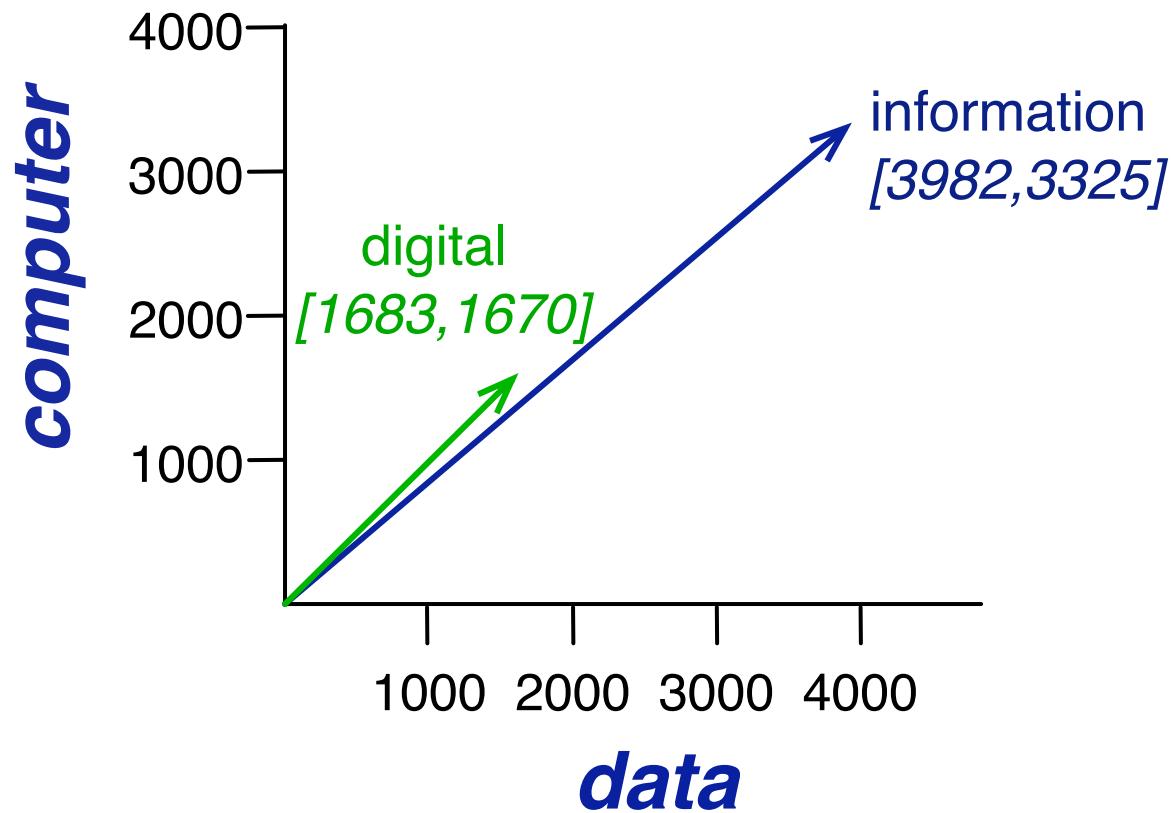
Un documento o una finestra di parole (per es. ±4) attorno alla parola data

- Due parole sono simili se i loro context vector lo sono

is traditionally followed by **cherry** pie, a traditional dessert
often mixed, such as **strawberry** rhubarb pie. Apple pie
computer peripherals and personal **digital** assistants. These devices usually
a computer. This includes **information** available on the internet

	aardvark	...	computer	data	result	pie	sugar	...
cherry	0	...	2	8	9	442	25	...
strawberry	0	...	0	0	1	60	19	...
digital	0	...	1670	1683	85	5	4	...
information	0	...	3325	3982	378	5	13	...

Esempio di visualizzazione dei context vector



Calcoliamo la similarità tra parole: prodotto scalare e similarità coseno

- Il prodotto scalare tra due vettori è uno scalare:

$$\text{dot product}(\mathbf{v}, \mathbf{w}) = \mathbf{v} \cdot \mathbf{w} = \sum_{i=1}^N v_i w_i = v_1 w_1 + v_2 w_2 + \dots + v_N w_N$$

- Tende a crescere quando i due vettori hanno grandi valori lungo le stesse dimensioni
- Può essere una buona metrica di similarità tra vettori

Problemi del prodotto scalare

- Il prodotto scalare favorisce i vettori lunghi infatti è tanto più elevato quanto il vettore è lungo cioè ha valori alti lungo molte dimensioni

$$|\mathbf{v}| = \sqrt{\sum_{i=1}^N v_i^2}$$

- Le parole frequenti come le stopwords (of, the, you) hanno vettori lunghi perché co-occorrono molte volte con le altre parole.
- Il prodotto scalare favorisce troppo le parole frequenti

La similarità coseno come alternativa

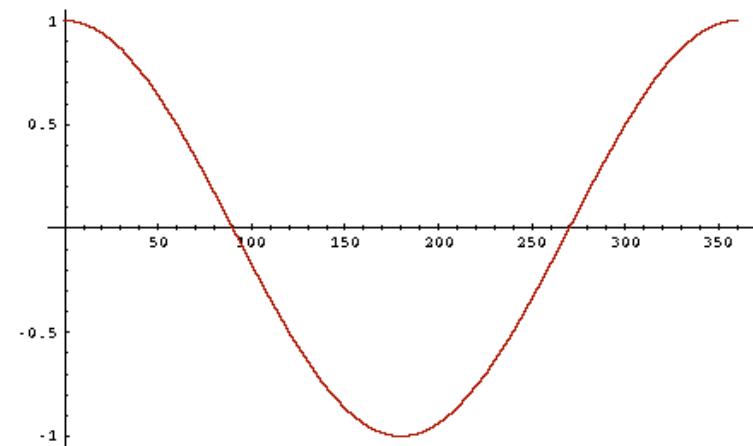
$$\text{cosine}(\vec{v}, \vec{w}) = \frac{\vec{v} \cdot \vec{w}}{|\vec{v}| |\vec{w}|} = \frac{\sum_{i=1}^N v_i w_i}{\sqrt{\sum_{i=1}^N v_i^2} \sqrt{\sum_{i=1}^N w_i^2}}$$

E' basata sulla definizione di prodotto scalare

$$\begin{aligned}\mathbf{a} \cdot \mathbf{b} &= |\mathbf{a}| |\mathbf{b}| \cos \theta \\ \frac{\mathbf{a} \cdot \mathbf{b}}{|\mathbf{a}| |\mathbf{b}|} &= \cos \theta\end{aligned}$$

Similarità coseno

- +1: per vettori che puntano **nella stessa direzione**
- -1: per vettori che puntano **in direzioni opposte**
- 0: per vettori **ortogonali**



Tuttavia, poiché i valori di frequenza grezzi sono non negativi, il coseno per questi vettori varia effettivamente da 0 a 1

Esempi

$$\cos(\vec{v}, \vec{w}) = \frac{\vec{v} \cdot \vec{w}}{\|\vec{v}\| \|\vec{w}\|} = \frac{\vec{v}}{\|\vec{v}\|} \cdot \frac{\vec{w}}{\|\vec{w}\|} = \frac{\sum_{i=1}^N v_i w_i}{\sqrt{\sum_{i=1}^N v_i^2} \sqrt{\sum_{i=1}^N w_i^2}}$$

	pie	data	computer
cherry	442	8	2
digital	5	1683	1670
information	5	3982	3325

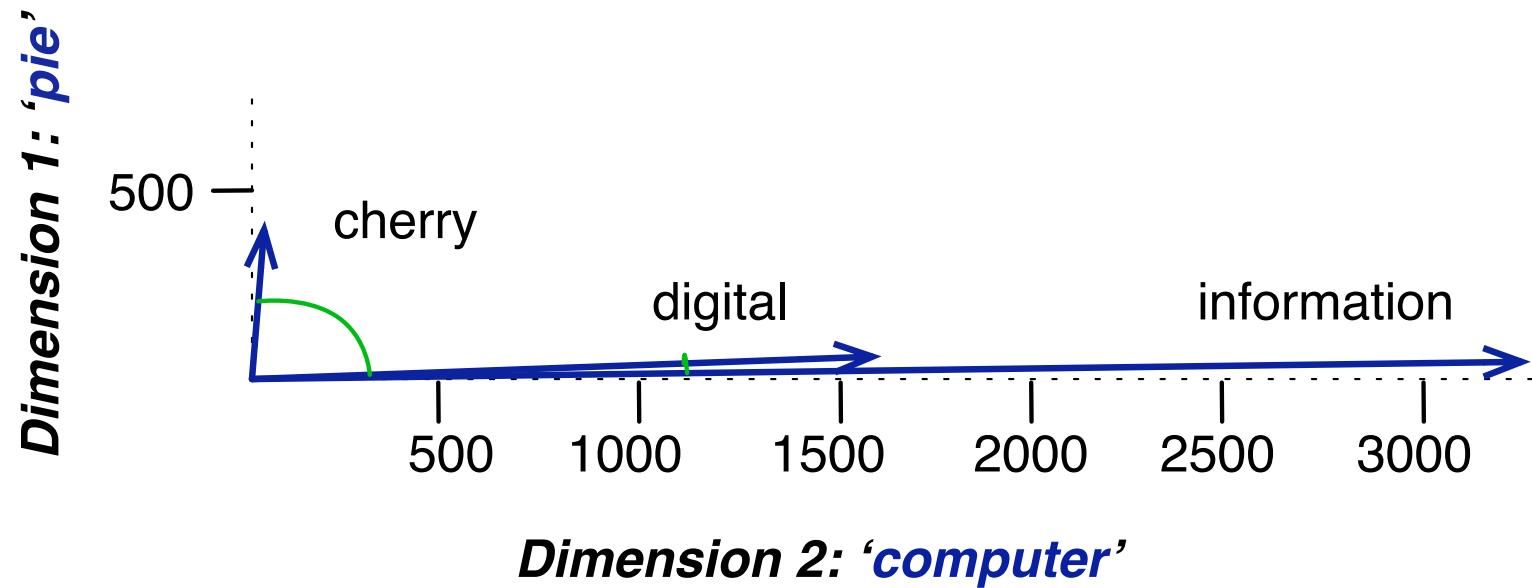
$$\cos(\text{cherry}, \text{information}) =$$

$$\frac{442 * 5 + 8 * 3982 + 2 * 3325}{\sqrt{442^2 + 8^2 + 2^2} \sqrt{5^2 + 3982^2 + 3325^2}} = .017$$

$$\cos(\text{digital}, \text{information}) =$$

$$\frac{5 * 5 + 1683 * 3982 + 1670 * 3325}{\sqrt{5^2 + 1683^2 + 1670^2} \sqrt{5^2 + 3982^2 + 3325^2}} = .996$$

Visualizziamo i coseni o meglio gli “angoli”



La frequenza grezza non è una buona rappresentazione

- Le matrici di co-occorrenza che abbiamo visto finora rappresentano ogni cella tramite le **frequenze delle parole**.
- La frequenza è certamente utile: se *sugar* compare spesso vicino a *apricot*, questa è un'informazione rilevante.
- Tuttavia, parole eccessivamente frequenti come *the*, *it* o *they* non sono molto informative riguardo al contesto.
- come possiamo bilanciare questi due vincoli in conflitto?

Due possibili soluzioni

- **tf-idf:** il valore di tf-idf per la parola t nel documento d è:

$$w_{t,d} = \text{tf}_{t,d} \times \text{idf}_t$$

Parole come "the" or "it" hanno idf molto bassa

- **PMI:** (Pointwise mutual information)

$$\text{PMI}(w_1, w_2) = \log \frac{p(w_1, w_2)}{p(w_1)p(w_2)}$$

Verifichiamo se parole come ad es. "good" appaiono molto spesso con "great" di quanto potremmo aspettarci per caso

Term frequency (tf)

- $\text{tf}_{t,d} = \text{count}(t,d)$
- Invece di usare direttamente il conteggio, applichiamo il logaritmo:

$$\text{tf}_{t,d} = \log_{10}(\text{count}(t,d)+1) \quad \text{Sempre } \geq 0$$

Applichiamo una scala logaritmica al conteggio grezzo perché non è direttamente collegato al significato del documento.

Document frequency (df)

- df_t è il numero di documenti in cui appare t .
- Non è il conteggio totale della parola su tutti i documenti
- "*Romeo*" molto discriminativo per una sola opera di Shakespeare:

	Collection Frequency	Document Frequency
Romeo	113	1
action	113	31

Inverse document frequency (idf)

$$\text{idf}_t = \log_{10} \left(\frac{N}{\text{df}_t} \right)$$

Sempre >= 0

N è il numero totale di documenti
nella collezione

Word	df	idf
Romeo	1	1.57
salad	2	1.27
Falstaff	4	0.967
forest	12	0.489
battle	21	0.246
wit	34	0.037
fool	36	0.012
good	37	0
sweet	37	0

Cos'è un documento?

- Qualunque unità di testo
- Normalmente ci si riferisce ad un paragrafo come a un documento

Calcolo finale del tf-idf

$$w_{t,d} = \text{tf}_{t,d} \times \text{idf}_t$$

- Raw counts:

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	1	0	7	13
good	114	80	62	89
fool	36	58	1	4
wit	20	15	2	3

- tf-idf:

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	0.074	0	0.22	0.28
good	0	0	0	0
fool	0.019	0.021	0.0036	0.0083
wit	0.049	0.044	0.018	0.022

Pointwise Mutual Information

- **Pointwise mutual information:**

Gli eventi x e y occorrono più spesso che se fossero statisticamente indipendenti?

$$\text{PMI}(X, Y) = \log_2 \frac{P(x,y)}{P(x)P(y)} \quad MI(X, Y) = \sum_x \sum_y P(x, y) \log_2 \frac{P(x, y)}{P(x)P(y)} = \\ = \mathbb{E}_{x \sim X, y \sim Y} [PMI(x, y)]$$

Tra due parole: (Church & Hanks 1989)

le parole w1 e w2 co-occorrono di più che se fossero indipendenti?

$$\text{PMI}(word_1, word_2) = \log_2 \frac{P(word_1, word_2)}{P(word_1)P(word_2)}$$

Positive Pointwise Mutual Information

- PMI varia tra $-\infty$ e $+\infty$
- I valori negativi sono problematici
 - $\text{PMI} < 0$: la co-occorrenza è **meno** di quanto ci aspetteremmo per caso
 - Inaffidabile senza corpora enormi
- I valori negativi di PMI vengono sostituiti da 0
- Positive PMI (**PPMI**) between word1 and word2:

$$\text{PPMI}(\text{word}_1, \text{word}_2) = \max\left(\log_2 \frac{P(\text{word}_1, \text{word}_2)}{P(\text{word}_1)P(\text{word}_2)}, 0\right)$$

Calcoliamo la PPMI su una term-context matrix

- Matrice F con W righe (parole) e C colonne (contesti)
- f_{ij} è il numero di volte in cui w_i occorre nel contesto c_j

$$p_{ij} = \frac{f_{ij}}{\sum_{i=1}^W \sum_{j=1}^C f_{ij}} \quad p_{i^*} = \frac{\sum_{j=1}^C f_{ij}}{\sum_{i=1}^W \sum_{j=1}^C f_{ij}} \quad p_{*j} = \frac{\sum_{i=1}^W f_{ij}}{\sum_{i=1}^C \sum_{j=1}^C f_{ij}}$$

	computer	data	result	pie	sugar	count(w)
cherry	2	8	9	442	25	486
strawberry	0	0	1	60	19	80
digital	1670	1683	85	5	4	3447
information	3325	3982	378	5	13	7703
count(context)	4997	5673	473	512	61	11716

$$pmi_{ij} = \log_2 \frac{p_{ij}}{p_{i^*} p_{*j}}$$

$$ppmi_{ij} = \begin{cases} pmi_{ij} & \text{if } pmi_{ij} > 0 \\ 0 & \text{otherwise} \end{cases}$$

$$p_{ij} = \frac{f_{ij}}{\sum_{i=1}^W \sum_{j=1}^C f_{ij}}$$

	computer	data	result	pie	sugar	count(w)
cherry	2	8	9	442	25	486
strawberry	0	0	1	60	19	80
digital	1670	1683	85	5	4	3447
information	3325	3982	378	5	13	7703
count(context)	4997	5673	473	512	61	11716

- $p(w=information, c=data) = 3982/11716 = .3399$
- $p(w=information) = 7703/11716 = .6575$
- $p(c=data) = 5673/11716 = .4842$

$$p(w_i) = \frac{\sum_{j=1}^C f_{ij}}{N} \quad p(c_j) = \frac{\sum_{i=1}^W f_{ij}}{N}$$

	p(w,context)					p(w)
	computer	data	result	pie	sugar	p(w)
cherry	0.0002	0.0007	0.0008	0.0377	0.0021	0.0415
strawberry	0.0000	0.0000	0.0001	0.0051	0.0016	0.0068
digital	0.1425	0.1436	0.0073	0.0004	0.0003	0.2942
information	0.2838	0.3399	0.0323	0.0004	0.0011	0.6575
p(context)	0.4265	0.4842	0.0404	0.0437	0.0052	

$$pmi_{ij} = \log_2 \frac{p_{ij}}{p_{i^*} p_{*j}}$$

	p(w,context)					p(w)
	computer	data	result	pie	sugar	p(w)
cherry	0.0002	0.0007	0.0008	0.0377	0.0021	0.0415
strawberry	0.0000	0.0000	0.0001	0.0051	0.0016	0.0068
digital	0.1425	0.1436	0.0073	0.0004	0.0003	0.2942
information	0.2838	0.3399	0.0323	0.0004	0.0011	0.6575
p(context)	0.4265	0.4842	0.0404	0.0437	0.0052	

- $pmi(\text{information}, \text{data}) = \log_2 (.3399 / (.6575 * .4842)) = .0944$

Resulting PPMI matrix (negatives replaced by 0)

	computer	data	result	pie	sugar
cherry	0	0	0	4.38	3.30
strawberry	0	0	0	4.10	5.51
digital	0.18	0.01	0	0	0
information	0.02	0.09	0.28	0	0

Weighting PMI

- PMI polarizzata verso gli eventi infrequenti
 - Parole molto rare hanno valori di PMI molto elevati
- Due soluzioni:
 - Dare alle parole rare probabilità un po' più alte
 - Usare lo smoothing add-1 che ha un effetto simile

Pesare la PMI

- Elevare le probabilità di contesto ad $\alpha = 0.75$:

$$\text{PPMI}_\alpha(w, c) = \max\left(\log_2 \frac{P(w, c)}{P(w)P_\alpha(c)}, 0\right)$$

$$P_\alpha(c) = \frac{\text{count}(c)^\alpha}{\sum_c \text{count}(c)^\alpha}$$

- Ci aiuta perché $P_\alpha(c) > P(c)$ per c raro
- Facciamo un esempio con $P(a) = .99$ and $P(b) = .01$
- $P_\alpha(a) = \frac{.99^{.75}}{.99^{.75} + .01^{.75}} = .97$ $P_\alpha(b) = \frac{.01^{.75}}{.99^{.75} + .01^{.75}} = .03$

Vettori sparsi vs vettori densi

- I vettori tf-idf o PMI sono
 - **lunghi** (dimensione $|V|$ tra 20.000 e 50.000)
 - **sparsi** (la maggior parte degli elementi sono nulli)
- In alternativa **apprendiamo** vettori che sono
 - **corti** (lunghezza tra 50 e 1000)
 - **densi** (la maggior parte degli elementi sono non nulli)

Vettori sparsi vs vettori densi

- Perché vettori densi e corti?

- Sono più facili da usare nei modelli di machine learning: ci sono meno parametri da apprendere
- Generalizzano meglio dei conteggi espliciti
- Catturano meglio la sinonimia
 - *car* e *automobile* sono sinonimi, ma sono dimensioni diverse
 - Una parola vche ha *car* come vicina e una che ha come vicina *automobile* potrebbero essere simili, ma in generale non lo sono

Word2vec

Metodi comuni per ottenere vettori corti e densi

- Ispirati dal NLP
 - Word2vec (skipgram, CBOW), GloVe
- Singular Value Decomposition (SVD)
 - Un caso speciale di SVD è la LSA (Latent Semantic Analysis)
- In alternativa a questi “embedding statici”:
 - Contextual Embeddings (ELMo, BERT)
 - Calcoliamo embedding distinti per la stessa parola in ogni suo contesto
 - Calcoliamo embedding idstinti per ogni token della parola

Semplici embedding statici che si possono scaricare

- Word2vec (Mikolov et al)
- <https://code.google.com/archive/p/word2vec/>

- GloVe (Pennington, Socher, Manning)
- <http://nlp.stanford.edu/projects/glove/>

Word2vec

- Metodo di embedding molto popolare
 - Molto veloce da addestrare
 - Codice disponibile sul web
- Idea: *predire* invece di contare
- Word2vec offre varie opzioni
Noi useremo: *skip-gram with negative sampling (SGNS)*

Word2vec

- Invece di contare quante volte ogni parola w compare vicino ad «albicocca» addestriamo un classificatore su un task di predizione binaria:
 - È probabile che w compaia vicino ad "albicocca"?
- In realtà non ci interessa questo task, ma useremo i pesi appresi dal classificatore come word embedding
- Grande idea: ***self-supervision***
 - Una parola c che compare vicino ad "albicocca" nel corpus funge da "risposta corretta" (gold standard) per l'apprendimento supervisionato
 - Non c'è bisogno di etichette umane

Bengio et al. (2003); Collobert et al. (2011)

Approccio: predire se una parola candidata c è un "vicino"

1. Trattiamo la parola target t e una parola di contesto vicina c come *esempi positivi*
2. Campioniamo casualmente altre parole nel lessico per ottenere *esempi negativi*
3. Usiamo la regressione logistica per addestrare un classificatore a distinguere i due casi
4. Usiamo i pesi appresi come embedding

Dati di training per Skip-Gram

- Assumiamo una finestra di +/- 2 parole, data la frase di training:

...lemon, a [tablespoon of apricot jam, a] pinch...
c1 c2 c3 c4
[target]

Classificatore Skip-Gram

- assumendo una finestra di +/- 2 parole

...lemon, a [tablespoon of apricot jam, a] pinch...
c1 c2 [target] c3 c4

- **Obiettivo:** addestrare un classificatore che, data una coppia candidata (parola, contesto)

(apricot, jam)
(apricot, aardvark)

...

- Assegna ad ogni coppia una probabilità:

$$P(+|w, c)$$
$$P(-|w, c) = 1 - P(+|w, c)$$

La similarità è calcolata dal prodotto scalare

- Due vettori sono simili se hanno un prodotto scalare elevato
 - Il coseno è solo un prodotto scalare normalizzato
- Di conseguenza:
 - $\text{Similarità}(w, c) \propto w \cdot c$
- Dovremo normalizzare per ottenere una probabilità
 - anche il coseno non è una probabilità

Modificare i prodotti scalari in probabilità

- $\text{Similarità}(w, c) \propto w \cdot c$
- Per trasformarlo in una probabilità useremo la sigmoide dalla regressione logistica:

$$P(+|w, c) = \sigma(c \cdot w) = \frac{1}{1 + \exp(-c \cdot w)}$$

$$\begin{aligned} P(-|w, c) &= 1 - P(+|w, c) \\ &= \sigma(-c \cdot w) = \frac{1}{1 + \exp(c \cdot w)} \end{aligned}$$

Come il classificatore Skip-Gram calcola $P(+|w, c)$

$$P(+|w, c) = \sigma(c \cdot w) = \frac{1}{1 + \exp(-c \cdot w)}$$

- Questo vale per una parola di contesto, ma abbiamo molte parole di contesto.
- Assumeremo l'indipendenza statisticae semplicemente le moltiplicheremo:

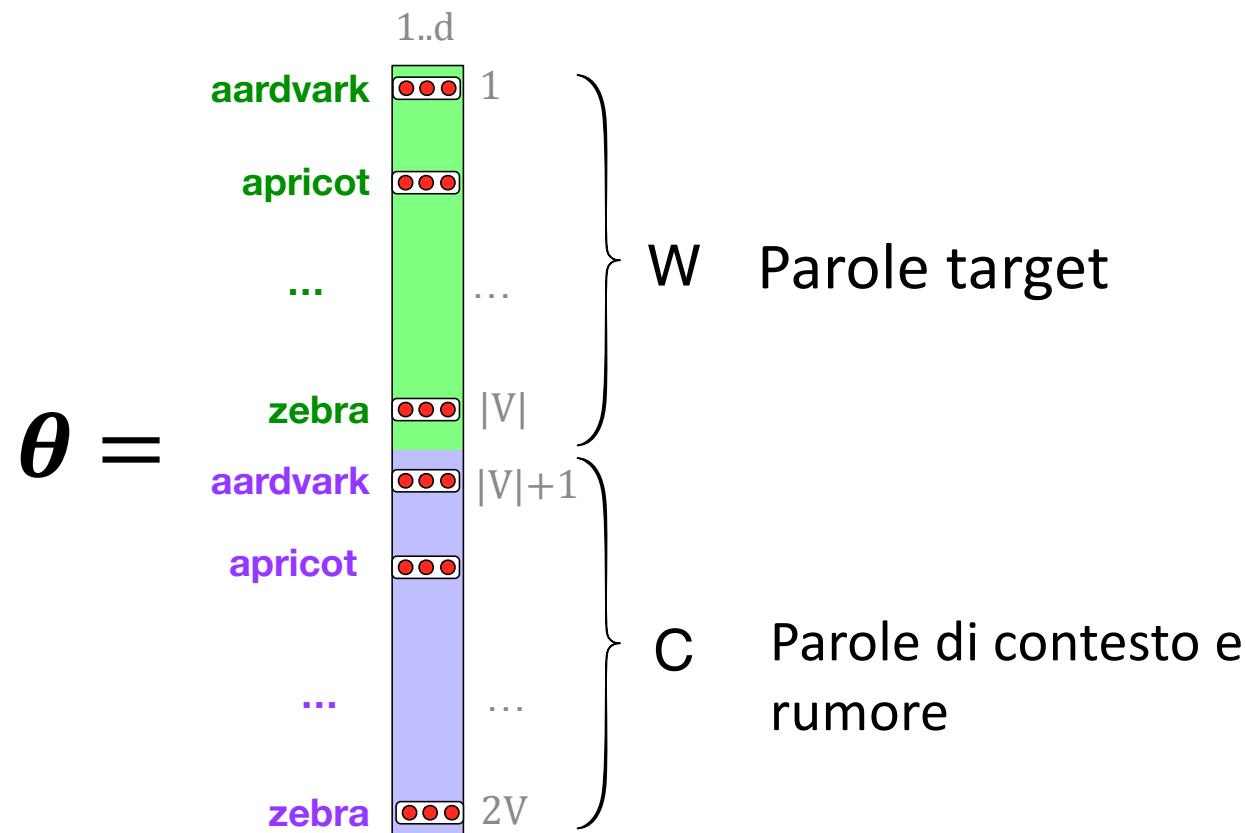
$$P(+|w, c_{1:L}) = \prod_{i=1}^L \sigma(c_i \cdot w)$$

$$\log P(+|w, c_{1:L}) = \sum_{i=1}^L \log \sigma(c_i \cdot w)$$

Riassumendo

- Un classificatore probabilistico che, data una parola target di test w e la sua finestra di contesto di L parole $c_{1:L}$
- Stima la probabilità che w compaia in questa finestra basandosi sulla similarità degli embedding di w con gli embedding di $c_{1:L}$.
- Per calcolare questo, abbiamo solo bisogno degli embedding per tutte le parole.

Avremo bisogno di un set di embedding per w ed uno per c



Dati di addestramento di Skip-Gram

...lemon, a [tablespoon of apricot jam, a] pinch...

c1 c2 [target] c3 c4



positive examples +

t c

apricot tablespoon

apricot of

apricot jam

apricot a

Dati di addestramento di Skip-Gram

...lemon, a [tablespoon of apricot jam, a] pinch...

c1 c2 [target] c3 c4



positive examples +

t c

apricot tablespoon

apricot of

apricot jam

apricot a

Per ogni esempio positivo, prenderemo k esempi negativi, campionando per frequenza

Dati di addestramento di Skip-Gram

...lemon, a [tablespoon of apricot jam, a] pinch...

c1 c2 [target] c3 c4



positive examples +

t	c
---	---

apricot	tablespoon
---------	------------

apricot	of
---------	----

apricot	jam
---------	-----

apricot	a
---------	---

negative examples -

t	c	t	c
---	---	---	---

apricot	aardvark	apricot	seven
---------	----------	---------	-------

apricot	my	apricot	forever
---------	----	---------	---------

apricot	where	apricot	dear
---------	-------	---------	------

apricot	coaxial	apricot	if
---------	---------	---------	----

Word2vec: come apprendere i vettori

- Dato l'insieme di istanze di training positive e negative, e un set iniziale di vettori di embedding
- L'obiettivo dell'apprendimento è aggiustare quei vettori di parole in modo da:
 - **Massimizzare** la similarità delle coppie (w, c_{pos}) estratte dai dati positivi
 - **Minimizzare** la similarità delle coppie (w, c_{neg}) estratte dai dati negativi.

Funzione di Loss per una parola w con $c_{pos}, c_{neg1} \dots c_{negk}$

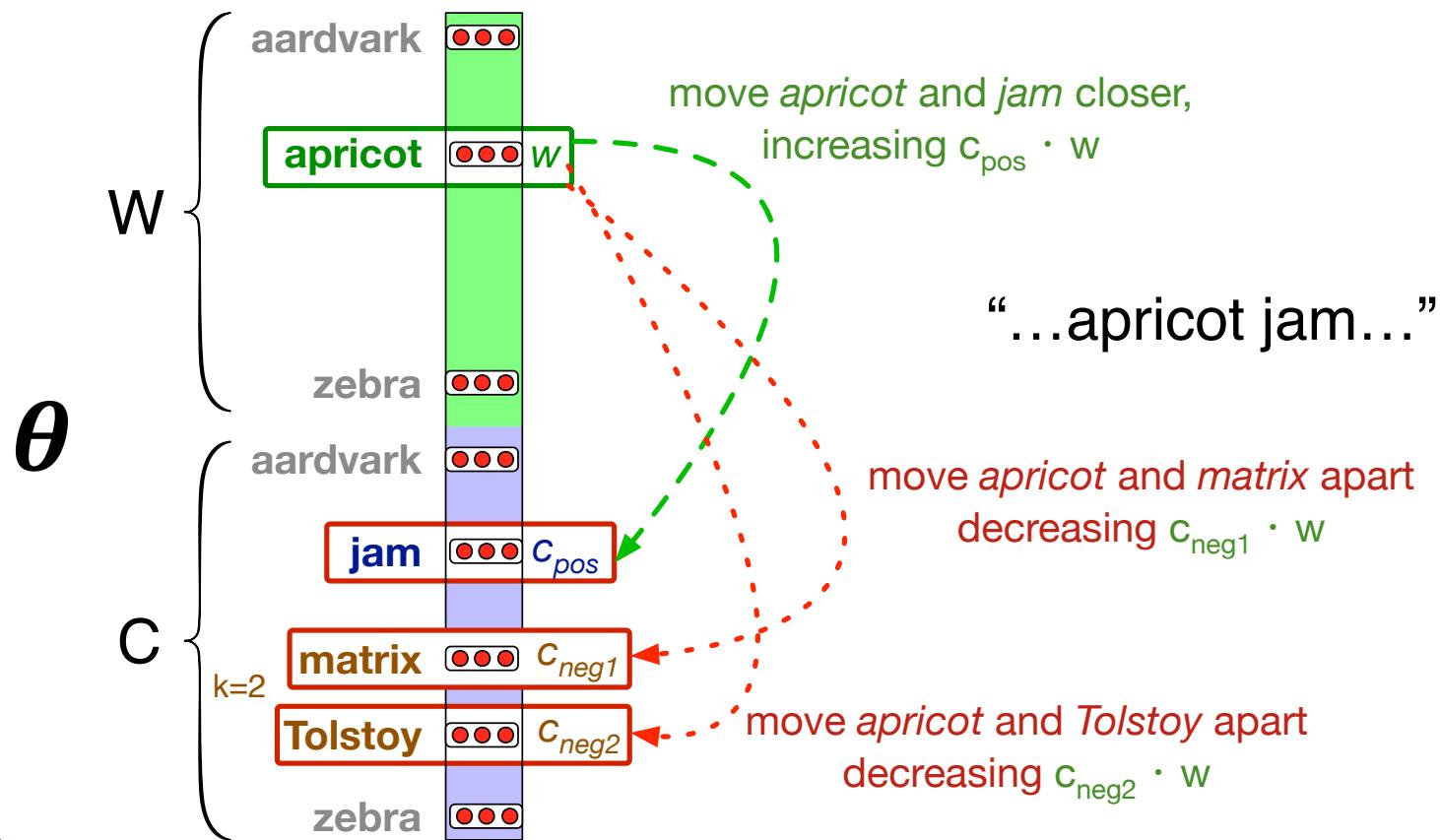
- Massimizzare la similarità del target con le parole di contesto reali, e minimizzare la similarità del target con le k parole non-vicine campionate negativamente.

$$\begin{aligned} L_{CE} &= -\log \left[P(+|w, c_{pos}) \prod_{i=1}^k P(-|w, c_{neg_i}) \right] \\ &= - \left[\log P(+|w, c_{pos}) + \sum_{i=1}^k \log P(-|w, c_{neg_i}) \right] \\ &= - \left[\log P(+|w, c_{pos}) + \sum_{i=1}^k \log (1 - P(+|w, c_{neg_i})) \right] \\ &= - \left[\log \sigma(c_{pos} \cdot w) + \sum_{i=1}^k \log \sigma(-c_{neg_i} \cdot w) \right] \end{aligned}$$

Apprendimento del classificatore

- Come apprendiamo?
Stochastic gradient descent!
- Troveremo il set di pesi per rendere le coppie positive più probabili e quelle negative meno probabili sull'intero training set.

Singolo passo del Gradient Descent



Gradient descent

- Ad ogni passo:
 - **Direzione:** Ci muoviamo nella direzione opposta al gradiente della funzione di perdita
 - **Magnitudine:** Muoviamo il valore di questo gradiente $\frac{d}{dw} L(f(x; w), y)$ pesato da un **learning rate** η
 - Un learning rate più alto significa muovere w più velocemente

$$w^{t+1} = w^t - \eta \frac{d}{dw} L(f(x; w), y)$$

Derivate della funzione di Loss

$$L_{CE} = - \left[\log \sigma(c_{pos} \cdot w) + \sum_{i=1}^k \log \sigma(-c_{neg_i} \cdot w) \right]$$

$$\frac{\partial L_{CE}}{\partial c_{pos}} = [\sigma(c_{pos} \cdot w) - 1]w$$

$$\frac{\partial L_{CE}}{\partial c_{neg}} = [\sigma(c_{neg} \cdot w)]w$$

$$\frac{\partial L_{CE}}{\partial w} = [\sigma(c_{pos} \cdot w) - 1]c_{pos} + \sum_{i=1}^k [\sigma(c_{neg_i} \cdot w)]c_{neg_i}$$

Aggiornamento in SGD

Iniziamo con le matrici C e W inizializzate casualmente, quindi eseguiamo aggiornamenti incrementali.

$$c_{pos}^{t+1} = c_{pos}^t - \eta [\sigma(c_{pos}^t \cdot w^t) - 1] w^t$$

$$c_{neg}^{t+1} = c_{neg}^t - \eta [\sigma(c_{neg}^t \cdot w^t)] w^t$$

$$w^{t+1} = w^t - \eta \left[[\sigma(c_{pos} \cdot w^t) - 1] c_{pos} + \sum_{i=1}^k [\sigma(c_{neg_i} \cdot w^t)] c_{neg_i} \right]$$

Due insiemi di embedding

- Il modello SGNS apprende ***due insiemi di embedding***:
 - la matrice degli embedding target (W)
 - la matrice degli embedding context (C)

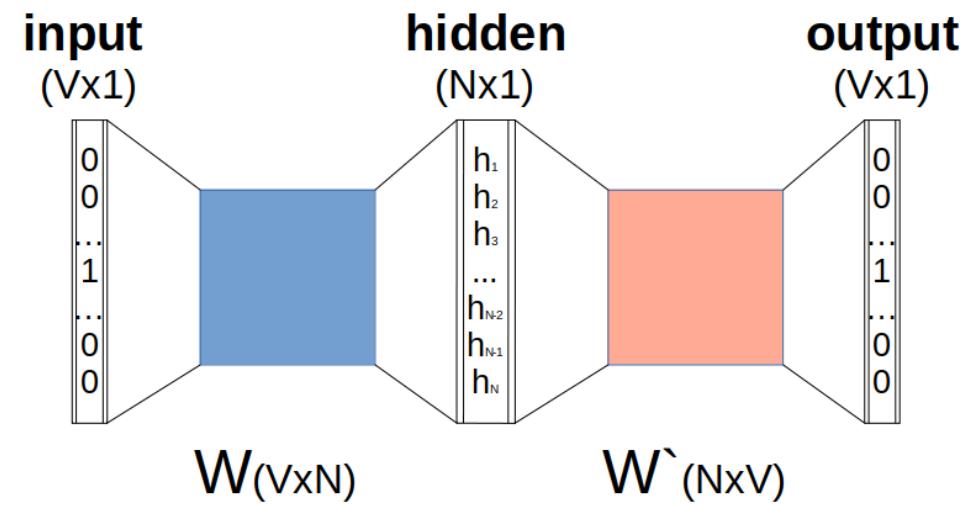
È pratica comune ***sommarli***, rappresentando la parola i con il vettore $w_i + c_i$.

Riassumendo: apprendimento degli embedding Word2vec Skip-gram

- Iniziamo con V vettori casuali d -dimensionali come embedding iniziali
 - Addestriamo un classificatore basato sulla similarità degli embedding
1. Prendiamo un corpus e consideriamo le coppie di parole che co-occorrono come esempi positivi
 2. Prendiamo coppie di parole che non co-occorrono come esempi negativi
 3. Addestriamo un classificatore a distinguerle, aggiustando lentamente tutti gli embedding per migliorare le prestazioni del classificatore
 4. Scartiamo il classificatore e teniamo gli embedding.

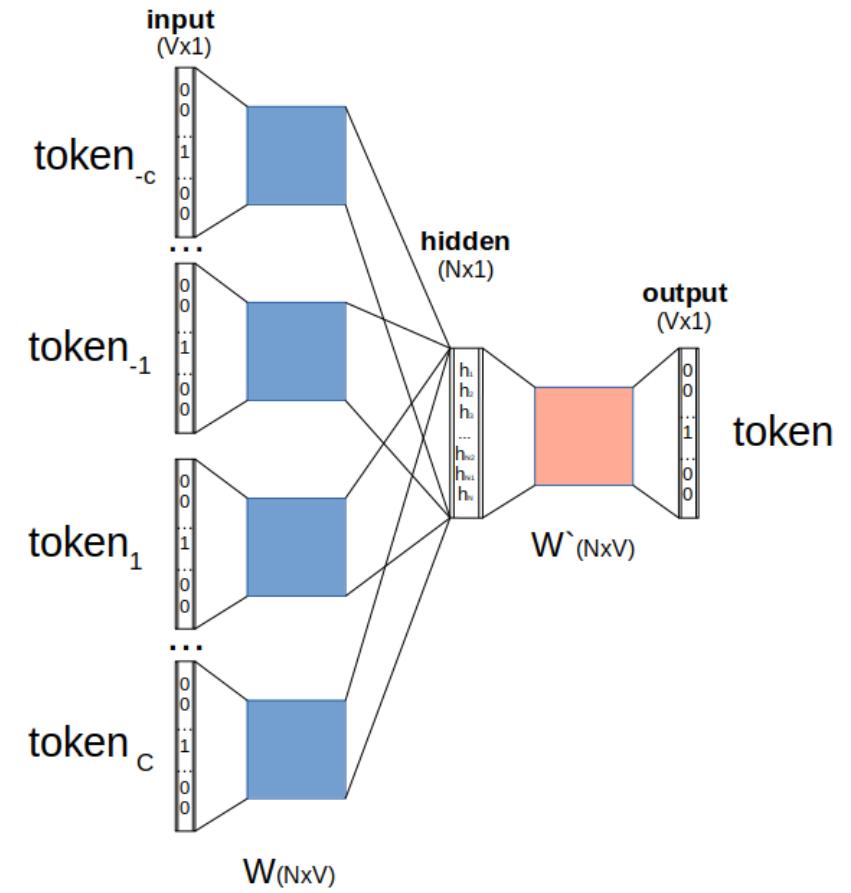
Skip-gram e CBOW come rete neurale

- Architettura fully-connected generica
- One-hot encoding per ogni parola
- Vogliamo apprendere la matrice W
- Lo strato nascosto ha dimensione $N \times 1$ per fornire la giusta dimensione a W



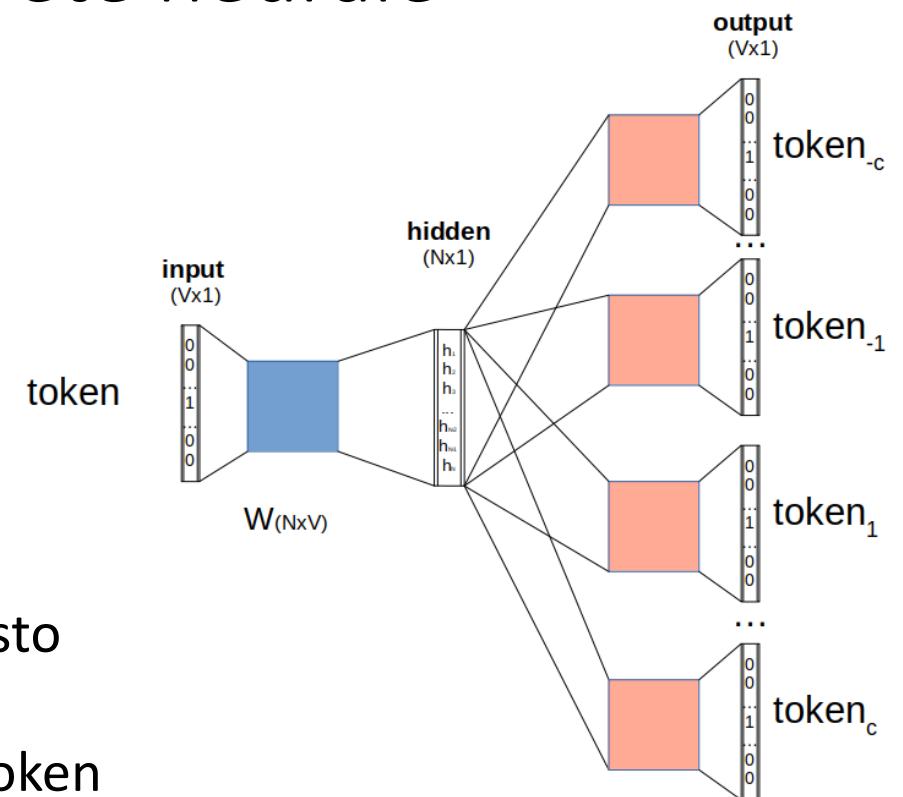
Skip-gram e CBOW come rete neurale

- CBOW – Context Bag Of Words
- Pediciamo la parola dal contesto
- La stessa matrice W per ogni token
- La riga W_i è l'embedding dell' i -esimo token
- Lo strato nascosto esegue una media degli ingressi, e quindi *perdiamo il positional encoding*



Skip-gram e CBOW come rete neurale

- Skip-gram
- Prediciamo il contesto dalla parola
- Un'unica matrice W
- Gli errori sulle uscite dei singoli token vengono sommati prima della backpropagation verso lo strato nascosto
- Lo strato nascosto è l'embedding del token

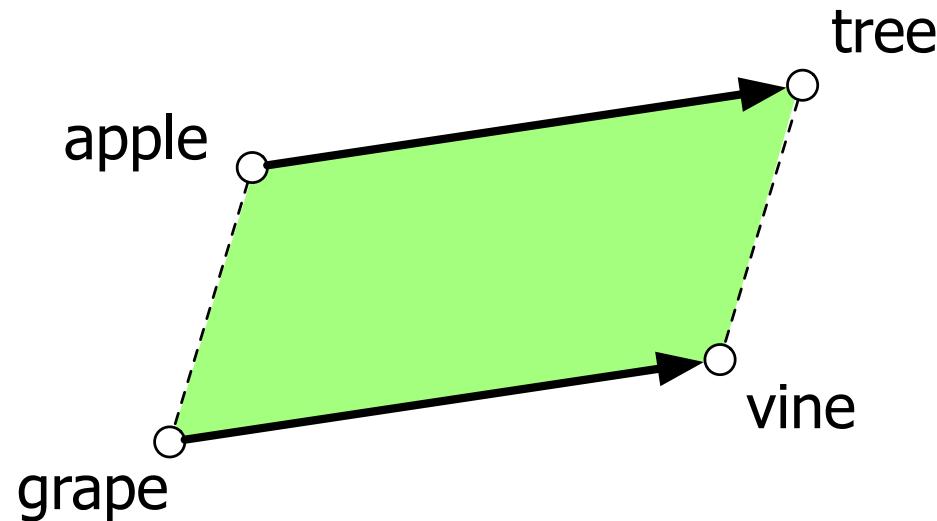


I tipi dei vicini dipendono dalla dimensione della finestra

- **Finestre piccole ($C = +/- 2$):** le parole più vicine sono parole sintatticamente simili nella stessa tassonomia
 - I vicini più prossimi di **Hogwarts** sono altre scuole fintizie: *Sunnydale, Evernight, Blandings*
- **Finestre grandi ($C = +/- 5$):** le parole più vicine sono parole correlate nello stesso campo semantico
 - I vicini più prossimi di **Hogwarts** sono parole del mondo di Harry Potter: *Silente, mezzosangue, Malfoy*

Relazioni di analogia

- Il modello classico del parallelogramma per il ragionamento per analogia (Rumelhart and Abrahamson 1973)
- Per risolvere: "*apple sta a tree come grape sta a _____*"
- *Sommiamo tree – apple a grape per ottenere vine*



Analogical relations via parallelogram

- Il metodo del paralleogramma può risolvere analogie sia con embedding sparsi sia con quelli densi (Turney and Littman 2005, Mikolov et al. 2013b)
 - king – man + woman è vicino a queen
 - Paris – France + Italy è vicino a Rome
-
- Per un problema $\mathbf{a} : \mathbf{b} :: \mathbf{a}^* : \mathbf{b}^*$, il metodo del paralleogramma calcola:

$$\hat{\mathbf{b}}^* = \underset{\mathbf{x}}{\operatorname{argmin}} \text{ distance}(\mathbf{x}, \mathbf{b} - \mathbf{a} + \mathbf{a}^*)$$

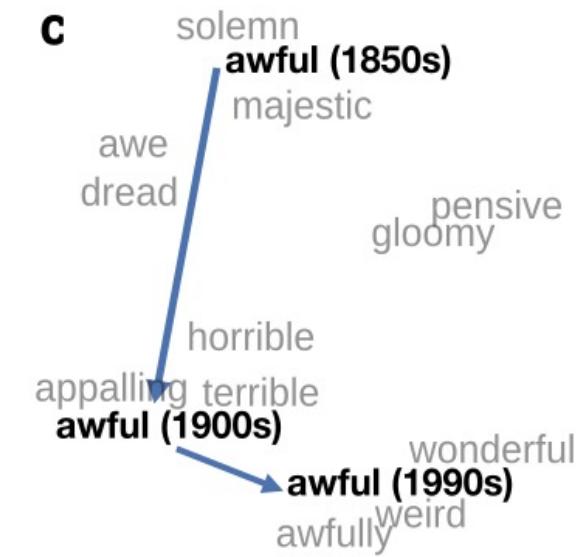
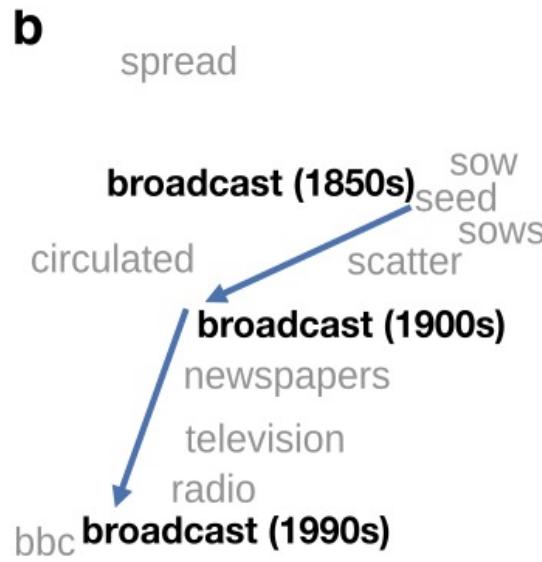
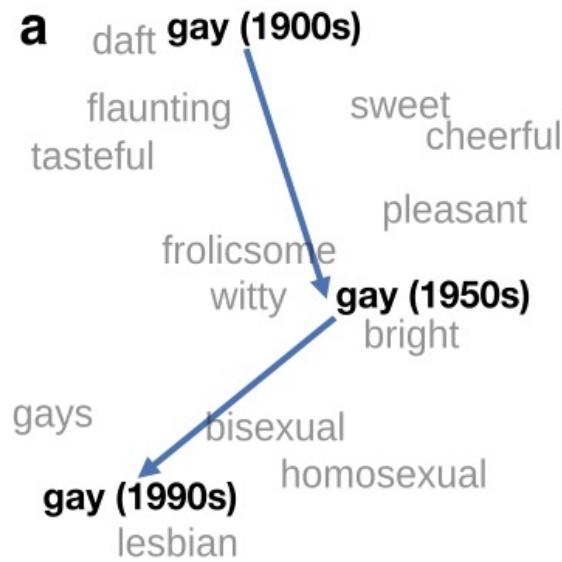
Avvertenze sul metodo del parallelogramma

- Sembra funzionare solo per parole frequenti, piccole distanze e certe relazioni (legare paesi a capitali, o parti del discorso), ma non per altre. (Linzen 2016, Gladkova et al. 2016, Ethayarajh et al. 2019a)
- Comprendere l'analogia è ancora un campo di ricerca aperto (Peterson et al. 2020)

Embedding come una finestra sulla semantica storica

Addestriamo gli embedding su decenni diversi di testi storici per osservare il cambiamento dei significati

~30 milioni di libri dal 1850 al 1990, dati di Google Books



William L. Hamilton, Jure Leskovec, and Dan Jurafsky. 2016. Diachronic Word Embeddings Reveal Statistical Laws of Semantic Change. Proceedings of ACL.

Gli embedding riflettono I bias culturali!

Bolukbasi, Tolga, Kai-Wei Chang, James Y. Zou, Venkatesh Saligrama, and Adam T. Kalai. "Man is to computer programmer as woman is to homemaker? debiasing word embeddings." In *NeurIPS*, pp. 4349-4357. 2016.

- Se chiediamo “Paris : France :: Tokyo : x”
 - x = Japan
- Se chiediamo “father : doctor :: mother : x”
 - x = nurse
- Se chiediamo “man : computer programmer :: woman : x”
 - x = homemaker

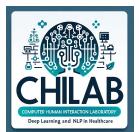
Algoritmi che usano embedding, ad esempio, nelle ricerche di personale per programmatore, potrebbero portare a bias nelle assunzioni

Fasttext e GloVe



Università
degli Studi
di Palermo

dipartimento
di ingegneria
unipa



fasttext (Bojanowski et al., 2017)

- word2vec non gestisce bene le parole sconosciute cioè quelle che appaiono nel test set, ma non nel training set
- Si parla di «word sparsity»
- Linguaggi con una morfologia ricca in cui alcune delle tante forme dei nomi e dei verbi occorrono raramente

fasttext

- [fasttext](#) gestisce questi problemi usando modelli a subword
- Ogni parola è rappresentata da sé stessa più una *bag* di n-grammi costituenti, cui vengono aggiunti i simboli speciali '<' e '>' per indicare le terminazioni
- $n = 3$, where →
 $\langle \text{where} \rangle, \langle \text{wh}, \text{ whe}, \text{ her}, \text{ ere}, \text{ re} \rangle$

fasttext

- [fasttext](#) gestisce questi problemi usando modelli a subword
 - Ogni parola è rappresentata da sé stessa più una *bag* di n-grammi costituenti, cui vengono aggiunti i simboli speciali '<' e '>' per indicare le terminazioni
1. Viene appreso un embedding skip-gram per ogni n-gramma costituente e per la parola
 2. La parola è rappresentata dalla somma di tutti gli embedding dei suoi n-grammi costituenti

GloVe (Pennington et al., 2014)

- GloVe significa *Global Vectors*
- Il modello cerca di catturare le statistiche globali del corpus
- Un modello log-bilineare con una funzione obiettivo ai minimi quadrati pesati, addestrato sulle voci non nulle di una matrice globale di co-occorrenza parola-parola

GloVe (Pennington et al., 2014)

Probability and Ratio	$k = solid$	$k = gas$	$k = water$	$k = fashion$
$P(k ice)$	1.9×10^{-4}	6.6×10^{-5}	3.0×10^{-3}	1.7×10^{-5}
$P(k steam)$	2.2×10^{-5}	7.8×10^{-4}	2.2×10^{-3}	1.8×10^{-5}
$P(k ice)/P(k steam)$	8.9	8.5×10^{-2}	1.36	0.96

*Le parole più discriminative
hanno probabilità alte →
rapporto >> 1*

*Le parole non discriminative
hanno circa la stessa
probabilità → rapporto ~ 1*

GloVe (Pennington et al., 2014)

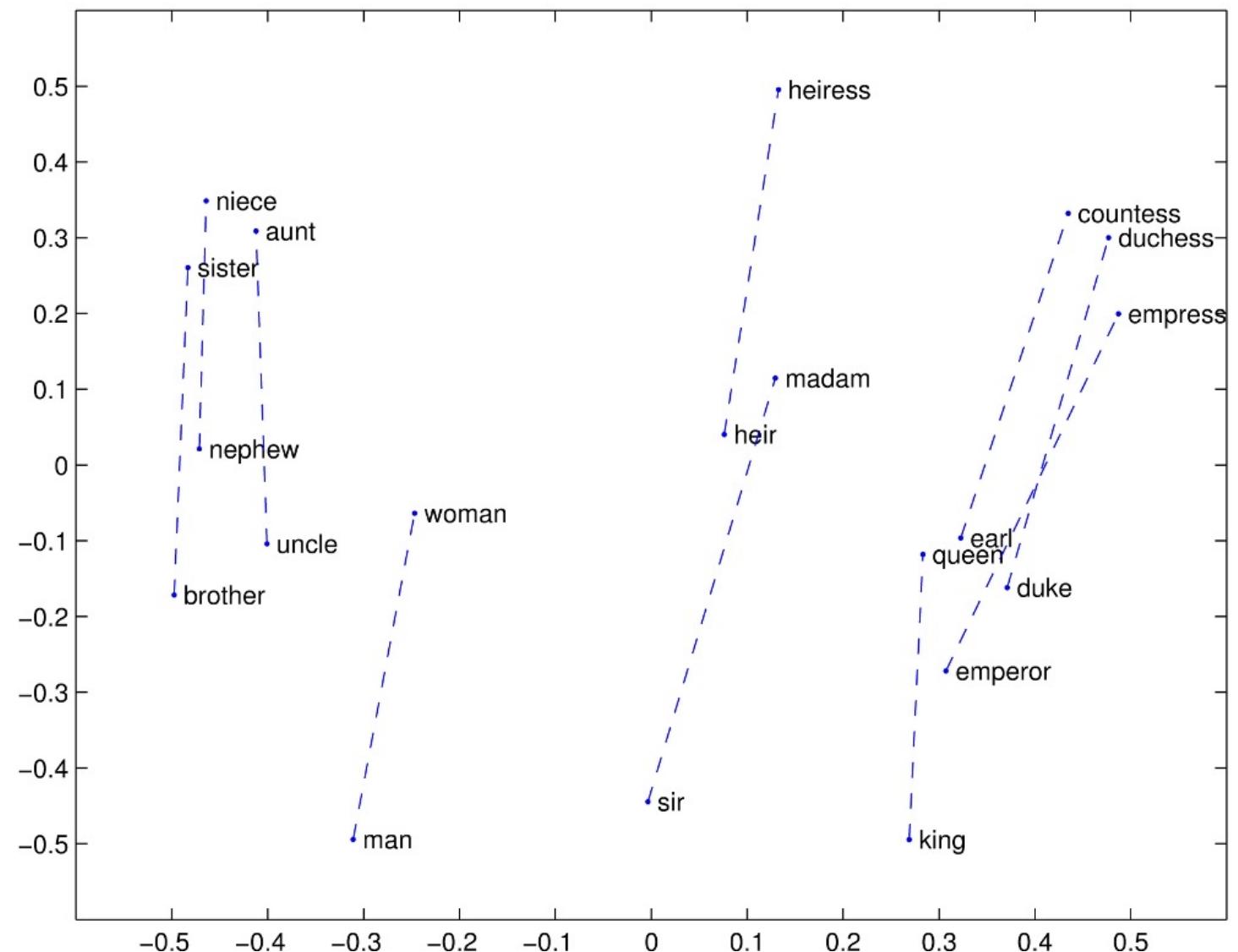
- GloVe apprende i vettori w_i in modo tale che

$$w_i \cdot w_j \approx \log P(w_i | w_j)$$

- Ne consegue che:

$$\log \frac{P(w_i | w_j)}{P(w_i | w_k)} = \log P(w_i | w_j) - \log P(w_i | w_k) \approx w_i \cdot (w_j - w_k)$$

Struttura dello spazio degli embedding GloVe

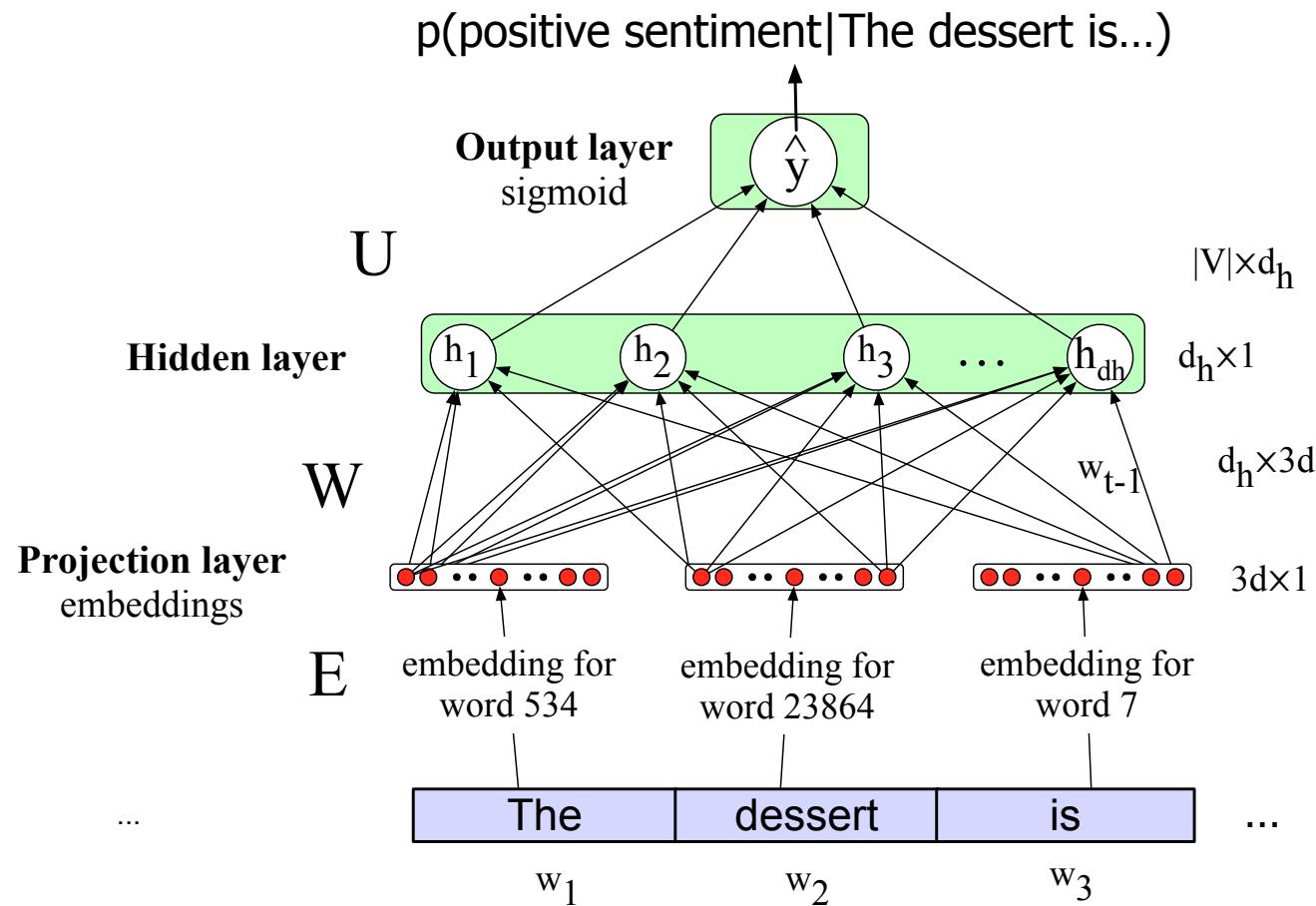


Language Model Neurali

Language Model neurali

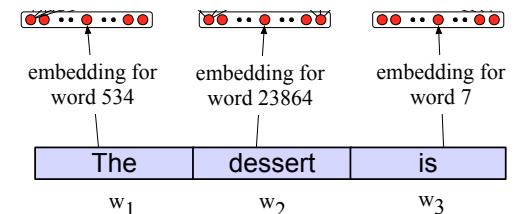
- Language Modeling: calcolare la probabilità di una parola in una sequenza, posto di averne osservate alcune.
- Abbiamo visto gli N-grammi
- I LM neurali li battono di gran lunga
- I LM neurali allo stato dell'arte sono basati su reti neurali molto potenti quali i Transformer
- Anche semplici modelli feedforward hanno una buona performance

Esempio: classificatore neurale con embedding come feature di ingresso



Problema: I testi hanno dimensioni differenti

- Il nostro esempio assume lunghezza fissa pari a tre
- Irrealistico
- Alcune semplici soluzioni
 1. Creiamo un input lungo quanto la più lunga sequenza osservata
 - Pad con 0 per sequenze più corte
 - Tronchiamo le sequenze di test più lunghe dell'input
 2. Creiamo un unico “sentence embedding” della stessa dimensionalità dei word embedding
 - Consideriamo la media di tutti i word embedding
 - Consideriamo il massimo elemento per elemento di tutti i word embedding
 - Il massimo per ogni dimensione tra tutte le parole

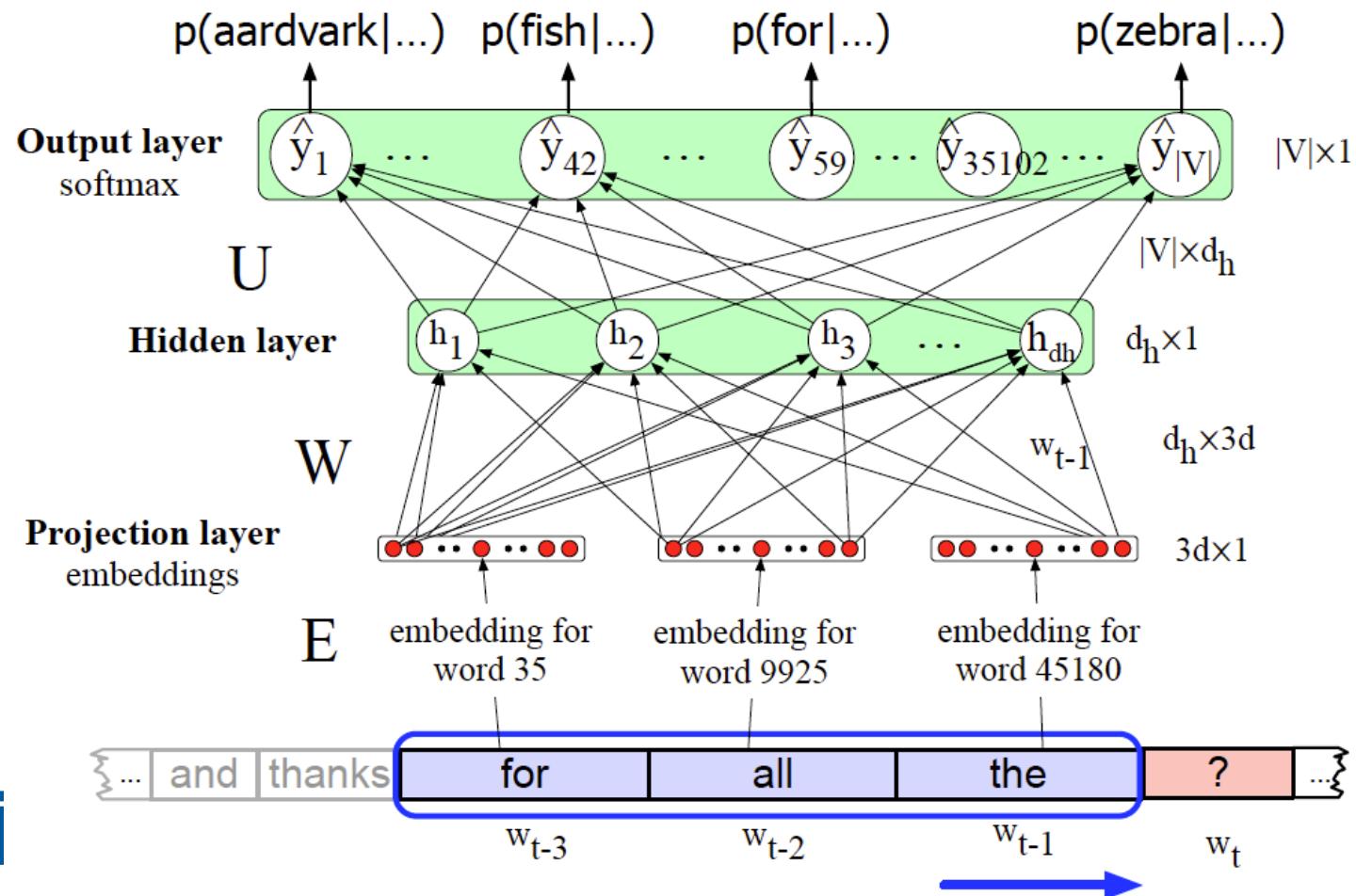


Semplice neural language model feedforward

- **Task:** predire la parola successiva w_t
- date le parole precedenti $w_{t-1}, w_{t-2}, w_{t-3}, \dots$
- **Problema:** dobbiamo gestire sequenze di lunghezza arbitraria.
- **Soluzione:** Sliding windows (di lunghezza fissa)

$$P(w_t | w_1^{t-1}) \approx P(w_t | w_{t-N+1}^{t-1})$$

Neural Language Model



Perché i LM neurali sono migliori degli N-grammi

- **Dati di training:**
 - Abbiamo osservato: Devo esser certo che il gatto abbia mangiato
 - Non abbiamo mai visto: il cane abbia mangiato
-
- **Dati di test:**
 - Ho scordato di assicurarmi che il cane _____
 - Il LM a N-grammi non può predire “abbia mangiato”!
 - Il LM neurale usa la similarità tra gli embedding di “gatto” e “cane” per generalizzare e predire “abbia mangiato” dopo cane