

MODULI FORMATIVI ESTATE 2021
INTELLIGENZA ARTIFICIALE: PROFESSIONE FUTURO

Natural Language Processing

Lezione IV - 2 settembre 2021



ARTIFICIAL INTELLIGENCE
& DATA ANALYTICS

Intelligenza

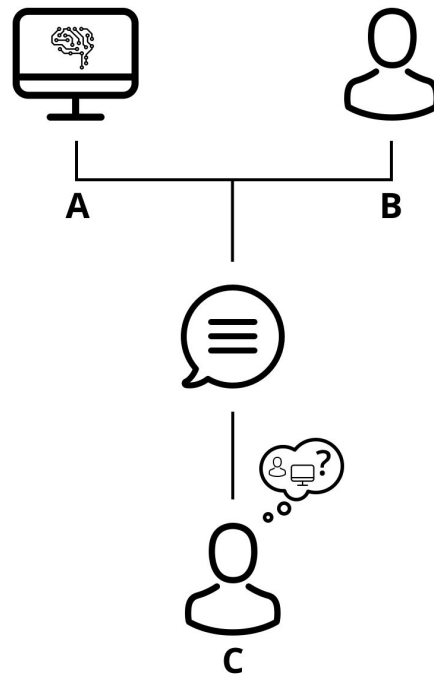
Quale è una definizione esatta e precisa di intelligenza?

Intelligenza Artificiale

- Non abbiamo una definizione ufficiale per “intelligenza”
- ... eppure parliamo di Intelligenza Artificiale (AI)
- Che cosa è una AI?
- Come possiamo capire se qualcosa di “artificiale” è anche “intelligente”?

Test di Turing

- Abbiamo tre attori, **A**, **B** e **C**
- C viene separato da **A** e **B**
- C deve capire se **A** e **B** sono persone o macchine
- Se una macchina (**A**) riesce ad ingannare **C**, allora **A** è **intelligente**



Natural Language Processing

- Natural Language (linguaggio naturale)
 - Come le persone comunicano tra di loro
- Processing
 - Come un calcolatore elabora le informazioni

Natural Language Processing

Come un calcolatore gestisce testi in linguaggio naturale

Cosa è NLP

Tecniche per il trattamento dell'informazione espressa in linguaggio naturale

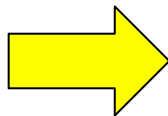
- **Tecniche sintattiche:** principalmente basate sul lessico e sulle distribuzioni statistiche dei termini
- **Tecniche grammaticali:** basate sulla struttura grammaticale o morfologica delle espressioni linguistiche
- **Tecniche semantiche:** basate sui significati dei termini e delle espressioni

Applicazioni di NLP

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ut eget efficitur lorem, et maximus massa. Fusce faucibus quis nunc sollicitudin commodo. Nam condimentum justo at purus malesuada tempor. Curabitur vitae lacus finibus, aliquet risus nec, aliquet felis. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Vestibulum ex felis, eleifend eu magna id, bibendum fermentum ante. Mauris sapien eros, interdum scelerisque egestas iaculis, eleifend sed erat. Donec eleifend lorem dictum eros imperdiet, interdum ultrices lorem tempus. Aenean id lacus eu ex eleifend mollis. Donec leo ex, tempus et eleifend id, sollicitudin sit amet leo. Maecenas in sem posuere nisl varius interdum feugiat cursus mauris. Nam ut bibendum elit. Pellentesque laoreet, orci tristique fringilla auctor, augue ipsum cursus turpis, in viverra odio ex eu diam. Nullam finibus arcu id mauris commodo faucibus. Aenean lobortis purus at sem scelerisque, vitae pellentesque arcu consectetur.

Cras imperdiet efficitur leo, sed dapibus sem porttitor ut. Vestibulum at neque turpis. Praesent tincidunt pretium urna, nec bibendum velit maximus ac. Quisque non mauris ultrices turpis malesuada faucibus. Suspendisse potenti. Mauris pretium tempor finibus. Suspendisse sit amet consequat dui. Maecenas luctus nunc libero, in viverra odio tristique ut. Sed sed tellus laoreet, rhoncus magna at, sagittis lectus. Sed quis elementum neque, quis pharetra dui.

Mauris non tincidunt quam. Aenean urna ligula, porttitor a purus a, convallis efficitur turpis. Duis tempus arcu est, ut imperdiet dolor tincidunt sit amet. Aliquam risus magna, dignissim vel consectetur in, porta eget ligula. Vivamus ullamcorper quis sem sit amet aliquet. Donec nec dolor rutrum, dictum libero sed, bibendum quam. In hac habitasse platea dictumst. Ut quis nunc sit amet quam dapibus finibus quis nec tellus. Sed porttitor vestibulum lacus nec semper. Vivamus odio velit, consequat sit amet auctor ut, pharetra eget mi. Suspendisse nec porttitor elit, in placerat justo. Nulla ac porttitor elit, vel sollicitudin odio.



Positive



Negative



Neutral

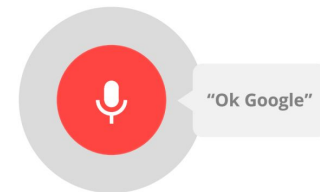
Sentiment Analysis

Applicazioni di NLP



ChatBot

Applicazioni di NLP



Hey Siri

Speech Recognition

Applicazioni di NLP

Μῆνιν ἄειδε θεὰ Πηληϊΐδαδω Ἀχιλλῆος
οὐλομένην, ἥ μυρὶν Ἀχαιοῖς ἄλγε' ἔθηκε,
πολλὰ δ' ἴφθιμου ψυχὰς Ἀτρεΐδην προΐαψε
ἥρωων, αὐτοῦ δ' ἑλῶρι' αὖτε ὕχεν κύνασσιν
σοῖωνοῖσιν τε παῖσιν, Διὸς δ' ἐτελεύετο βουλή,
ἔξ οὔ δὴ τὰ πρῶτα δικάστητην ἔρρισαντες



Cantami, o Diva, del Pelide Achille
L'ira funesta che infiniti addusse
Lutti agli Achei, molte anzi tempo all'Orco
Generose travolse alme d'eroi,
di cani e d'augelli orrido pasto
Lor salme abbandonò (così di Giove

L'alto consiglio s'adempia), da quando
Primamente disgiunse aspra contesa
Il re de' prodi Atride e il divo Achille.

Machine Translation

Applicazioni di NLP



Controllo ortografico



Ricerca parole chiave



Information Extraction



Advertising

Processare Informazioni

iris setosa



petal sepal

iris versicolor



petal sepal

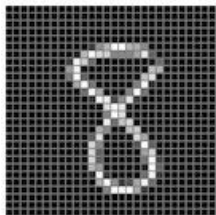
iris virginica



petal sepal

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa
5	6	5.4	3.9	1.7	0.4	Iris-setosa
6	7	4.6	3.4	1.4	0.3	Iris-setosa
7	8	5.0	3.4	1.5	0.2	Iris-setosa
8	9	4.4	2.9	1.4	0.2	Iris-setosa
9	10	4.9	3.1	1.5	0.1	Iris-setosa

Computer Vision

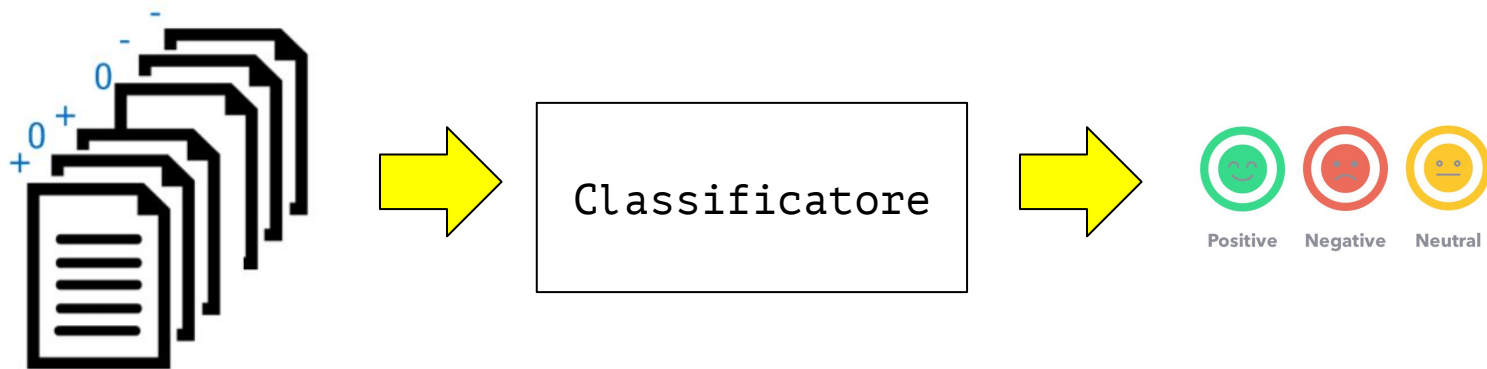


28 x 28
784 pixels

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

Sentiment Analysis

- Input: testo
- Output: classificazione **polarità** documento
 - Negativa
 - Neutra
 - Positiva



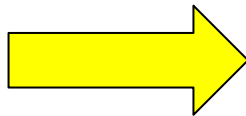
Che cosa andiamo a “processare”?

- Come elaboriamo il testo? Come lo rendiamo “digeribile” per un calcolatore?
- Ci serve una rappresentazione
 - Sequenza di lettere?
 - Sequenza di parole?
 - Sequenza di numeri?

Tokenization

Operazione per ridurre un testo in unità atomiche di nostra scelta (**parole**, caratteri, ...)

The quick brown fox
jumps over the lazy
dog

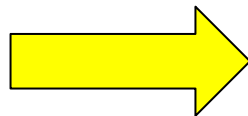


The
quick
brown
fox
jumps
over
the
lazy
dog

Tokenization

Operazione per ridurre un testo in unità atomiche di nostra scelta (parole, caratteri, ...)

The quick brown fox
jumps over the lazy
dog

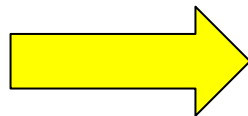


T	h	e		
q	u	i	c	k
b	r	o	w	n
f	o	x		
j	u	m	p	s
o	v	e	r	
t	h	e		
l	a	z	y	
d	o	g		

Tokenization

Operazione per ridurre un testo in unità atomiche di nostra scelta (parole, caratteri, **digrammi**)

The quick brown fox
jumps over the lazy
dog



Th	he	e			
q	qu	ui	ic	ck	k
b	br	ro	ow	wn	n
f	fo	ox	x		
j	ju	um	mp	ps	s
o	ov	ve	er	r	
t	th	he	e		
l	la	az	zy	y	
d	do	og	g		

n-grammi

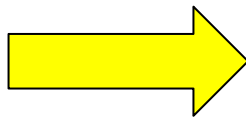
A volte può essere più interessante spezzare il testo in unità diversa da “lettera” o “parola”

- **n-gramma:** sottosequenza di lunghezza n di un testo
- Ci offre delle informazioni in più rispetto alle singole lettere
- Più flessibile che non usare parole intere (occupa meno spazio)
- Diversi usi, anche al di fuori di NLP: sequenziamento del DNA

“Bag of words”

Vettore che conta quante volte appare un token in un “documento”

The quick brown fox
jumps over the lazy
dog



The	2
quick	1
brown	1
fox	1
jumps	1
over	1
lazy	1
dog	1

Count vector per analizzare parole

Proviamo a cercare “affinità” tra le parole **gatto**, **matto** e **felino** contando quante parole hanno in comune su wikipedia

Gatto (<i>Felis silvestris catus</i>)		Matto (Il Matto)		Felino (Felidae)	
di	354	il	54	-	38
e	249	di	46	di	26
il	232	e	41	gatto	24
la	184	è	37	e	19
gatto	166	un	32	genere	16

Count vector per analizzare parole

Proviamo a cercare “affinità” tra le parole **gatto**, **matto** e **felino** contando quante parole hanno in comune su wikipedia

Gatto (<i>Felis silvestris catus</i>)		Matto (Il Matto)		Felino (Felidae)	
di	354	il	54	-	38
e	249	di	46	di	26
il	232	e	41	gatto	24
la	184	è	37	e	19
gatto	166	un	32	genere	16

Parole funzionali

Alcune parole sono molto frequenti, non hanno un significato proprio, ma hanno una funzione grammaticale.

Stop words:

- Non hanno un significato
- Di solito sono brevi (pochi caratteri)
- Modificano il significato di altre parole

Posso prendere una macchina

Posso prendere la macchina

“Bag of words” per analizzare parole

Proviamo a cercare “affinità” tra le parole **gatto**, **matto** e **felino** contando quante parole hanno in comune su wikipedia, eliminando le stopwords

Gatto (<i>Felis silvestris catus</i>)		Matto (Il Matto)		Felino (Felidae)	
gatto	166	==	20	-	38
gatti	74	matto	16	gatto	24
===	54	the	12	genere	16
==	32	può	12	==	12
può	31	altri	9	felidi	11

Può funzionare?

- “Bag of words” permette di risolvere qualche semplice problema
- ... ma per approcci più generali dovremmo usare vettori molto lunghi
- Ricordate: ogni elemento del vettore fa riferimento ad una parola del dizionario
- Occupa decisamente troppo spazio!
- Inoltre, parole che non sono nel vocabolario, non esistono!

Esercizio

Individuare Lingua

Dato un documento di testo, capire quale sia la lingua in cui è scritto

- Abbiamo a disposizione tanti esempi di testi in varie lingue
- Arriva un documento di cui non conosciamo la lingua
- Vogliamo automaticamente capire che lingua sia

Individuare Lingua

Dato un documento di testo, capire quale sia la lingua in cui è scritto

- Abbiamo a disposizione tanti esempi di testi in varie lingue
- Arriva un documento di cui non conosciamo la lingua
- Vogliamo automaticamente capire che lingua sia

Costruiamo un profilo per ogni lingua

Individuare la Lingua

Algoritmo per costruire un profilo:

- Spezziamo il testo in token rimuovendo punteggiatura
- Per ogni token costruiamo gli n-grammi (n da 1 a 5)
- Misuriamo quante volte compaiono i singoli n-grammi
- Teniamo i 300 n-grammi più frequenti

Profilo linguistico

Intuitivamente:

- Gli n-grammi più frequenti saranno gli 1-grammi
- Gli 1-grammi rispecchieranno le frequenze delle lettere in una lingua
- Subito dopo ci saranno gli n-grammi che rappresentano le stopwords di una lingua
- Oltre la 300^o posizione avremmo n-grammi relativi al contenuto

Individuare la Lingua

Algoritmo per assegnare una lingua ad un documento D:

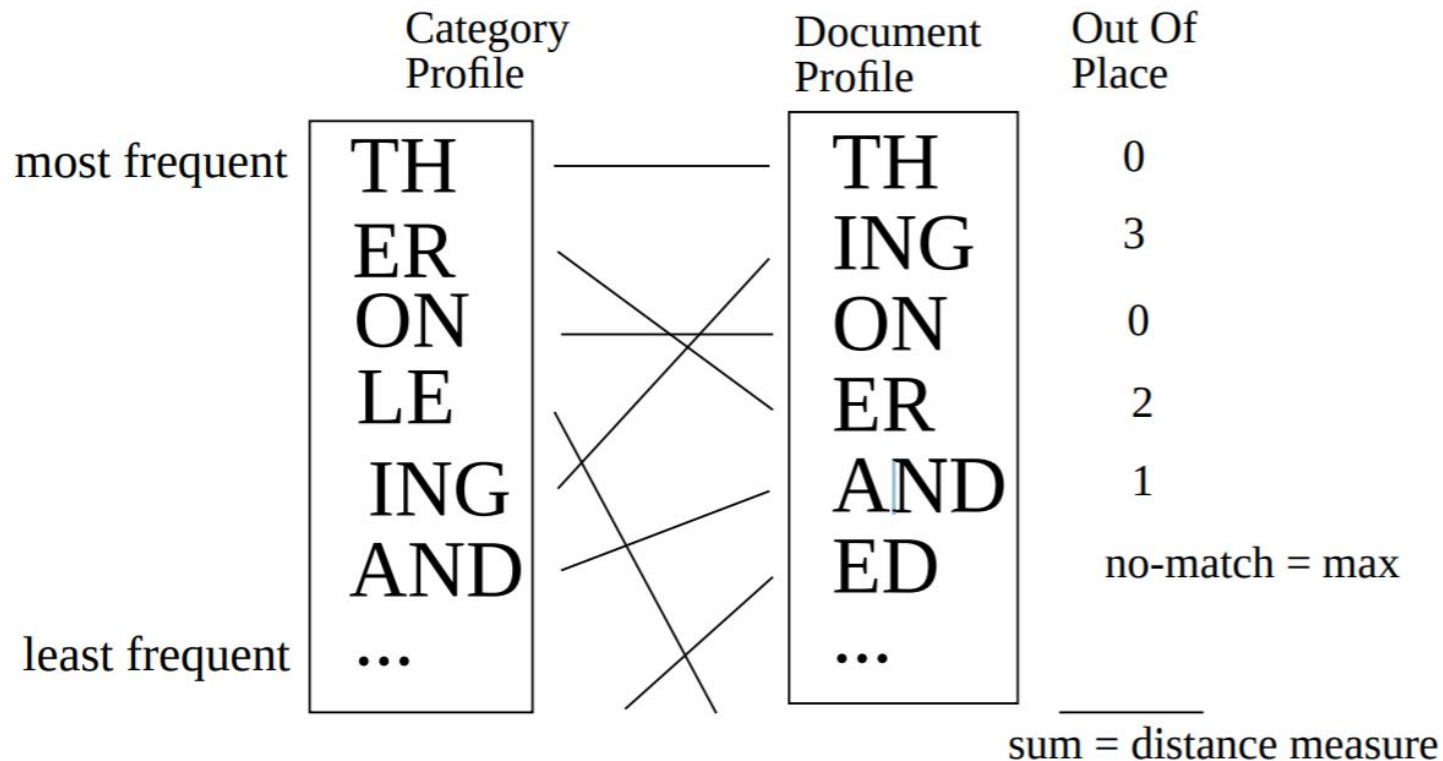
- Spezziamo il testo in token rimuovendo punteggiatura
- Per ogni token costruiamo gli n-grammi (n da 1 a 5)
- Misuriamo quante volte compaiono i singoli n-grammi
- Teniamo i 300 n-grammi più frequenti
- Misuriamo la distanza tra il profilo di D e quello di ogni lingua
- Il profilo a distanza minore sarà quello della lingua di D
- ... come misuriamo la distanza?

Individuare la lingua

Prendiamo due profili (quello del documento e quello di una lingua)

- Scorriamo gli n-grammi di un profilo dal primo all'ultimo
- Per ogni n-gramma, guardiamo se è nella stessa posizione nel secondo profilo
- Se è fuori posizione, contiamo di quante posizioni è “sfasato”
- Sommiamo tutte le differenze di posizione
- La somma finale sarà la distanza tra i profili!

Individuare la Lingua



Perdita di informazioni

Con il “bag of words” abbiamo una perdita di informazioni

Ordinamento

- ... good not necessarily ...
- ... not good ...

Contesto

- I clicked the mouse
- The cat caught the mouse

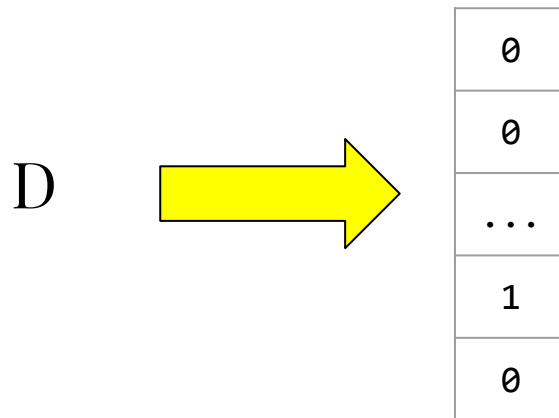
Perdita di informazioni

In una rappresentazione “bag of words” tutte le parole hanno la **stessa importanza**

Esistono altre rappresentazioni, che pesano ogni singola parola
(es: TF-IDF)

“Bag-of-words”: interpretazione

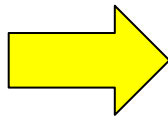
Abbiamo un documento D che contiene una **sola parola**



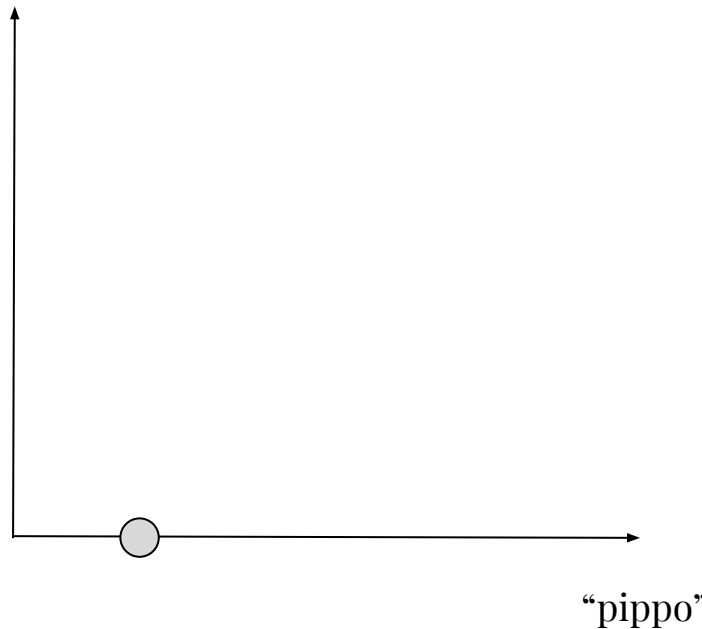
Interpretazione: “una parola è un punto di coordinate 1 sul proprio asse”

“Bag-of-words”: interpretazione

Pippo

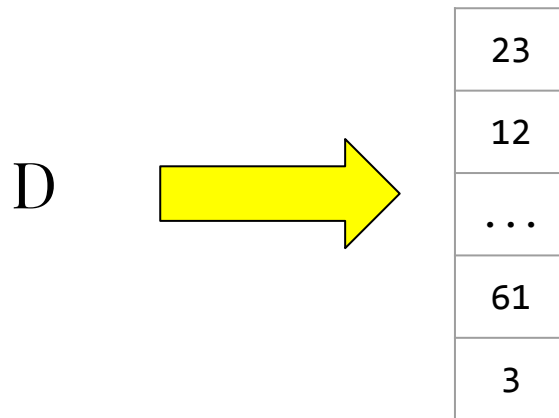


	0
	0
	0
	0
pippo	1
	0
	0
	0



“Bag-of-words”: interpretazione

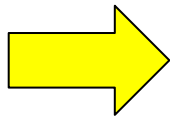
Abbiamo un documento D che contiene **molte parole**



Interpretazione: “un documento è il punto risultante dalla somma delle parole che contiene”

“Bag-of-words”: interpretazione

Chi è chi?



	0
chi	2
	0
	0
	0
	0
	0
è	1

“è”



“chi”

Problema di fondo

Con questa rappresentazione, abbiamo la stessa distanza tra ogni coppia di parole

- **Distanza** tra due parole con significato **simile** uguale a distanza tra due parole con significato **diverso**
- **Nessuna** informazione di carattere **semantico**
- Le parole **gatto**, **matto** e **felino** hanno la stessa distanza tra di loro

Distanze

Prendiamo dei documenti da wikipedia relativi a: Machine Learning, Artificial Intelligence, Soccer, Tennis

- Rappresentiamo i documenti come “bag of words”
- Vorremmo accadesse che
 - Documento ML vicino a documento AI
 - Documento ML lontano da documento Soccer
 - Documento ML lontano da documento Tennis
- Usiamo distanza euclidea

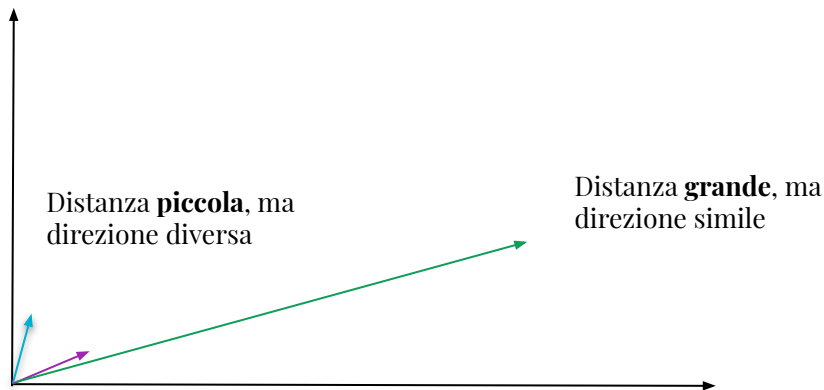
Distanze

- Distanza ML da AI: 661.10
- Distanza ML da Soccer: 459.30
- Distanza ML da Tennis: 805.40

... non funziona!

Motivo

- Direzione ci dice **quali** parole
- Norma ci dice **quante** parole
- Distanza tra due documenti dipende da **entrambe**



Soluzione

- Normalizzo: proietto ogni punto su una semicirconferenza
 - Norma sempre uguale, cambia solo direzione
- Misuro angolo, non distanza:
 - “cosine similarity”, 1 indica la similarità massima, 0 la minima

Distanza tra parole

Abbiamo visto come misurare una distanza tra documenti che hanno molte parole. Possiamo misurare distanza tra parole?

- Vettore “bag of words” con una sola parola
- Prende il nome di “one-hot”
- Tutte le parole hanno la stessa distanza tra di loro!
- **Nessuna** informazione di carattere **semantico**

Idea

Abbiamo uno spazio con:

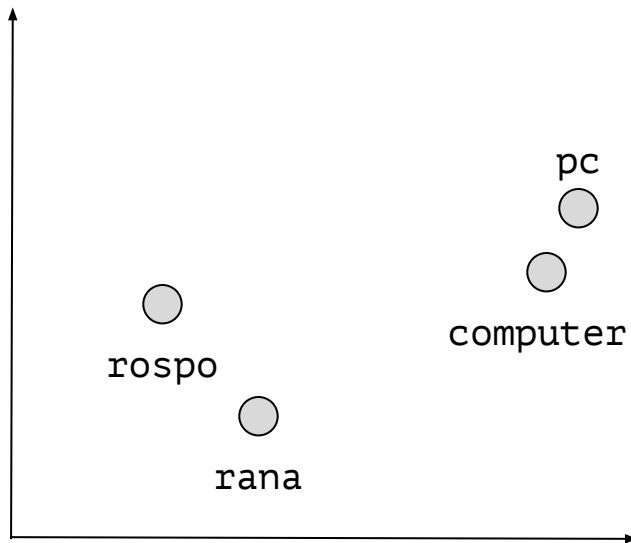
- Tante **dimensioni** (assi) quante le parole
- Distanza in questo spazio non hanno **valore semantico**

Vorremmo:

- Un numero minore di dimensioni
- Delle distanze con un **valore semantico**

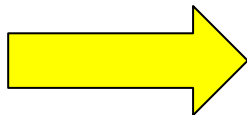
Nuova rappresentazione

Vogliamo costruire uno spazio dove parole con significato **simile** finiscano **vicine** tra di loro

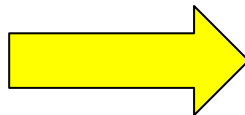


Nuova rappresentazione

rana



0
0
0
...
0
1
0



0.16
0.63
...
0.01
0.57

TF-IDF

Per ogni parola w in un documento D :

1. Quantifico “specificità” di w in D
 - a. w presente spesso \Rightarrow bassa
 - b. w presente raramente \Rightarrow alta
2. Peso w con la sua specificità in D

TF-IDF

1. Quantifico “specificità” di w in D
 - a. w presente spesso \Rightarrow bassa
 - b. w presente raramente \Rightarrow alta
2. Peso w con la sua specificità in D

Vettore D :

- Si allunga negli assi più specifici
- Si accorcia negli assi meno specifici

Cambia direzione complessiva: ruota verso gli assi più specifici

Specificità

Come si calcola la “specificità” (uno dei tanti modi):

- $\text{conta}(w,D)$ = numero di volte che compare w in D
- $\text{specificita}(w,D) = \log(|D| / \text{conta}(w,D))$

Se la parola è comune la specificità tende a 0, più la parola è rara più la specificità è alta

TF-IDF (nomi reali)

Come si calcola la “specificità” (uno dei tanti modi):

- $\text{conta}(w,D)$ = numero di volte che compare w in D

Term Frequency (TF)

- $\text{specificita}(w,D) = \log(|D| / \text{conta}(w,D))$

Inverse Document Frequency (IDF)

Word Embedding

Esistono delle rappresentazioni già pronte

- glove
- Word2vec
- ...

Si tratta di trasformazioni che proiettano le nostre parole nel nuovo spazio

Proprietà interessanti

Le parole più vicine a San Francisco:

- Los Angeles
- Golden Gate
- Oakland
- California
- San Diego
- Pasadena
- Seattle

Proprietà interessanti

Le parole più vicine a France:

- Spain
- Belgium
- Netherlands
- Italy
- Switzerland
- Portugal
- Russia

Proprietà interessanti

Cosa succede se “sommo” due parole? Si tratta di vettori, posso farlo.

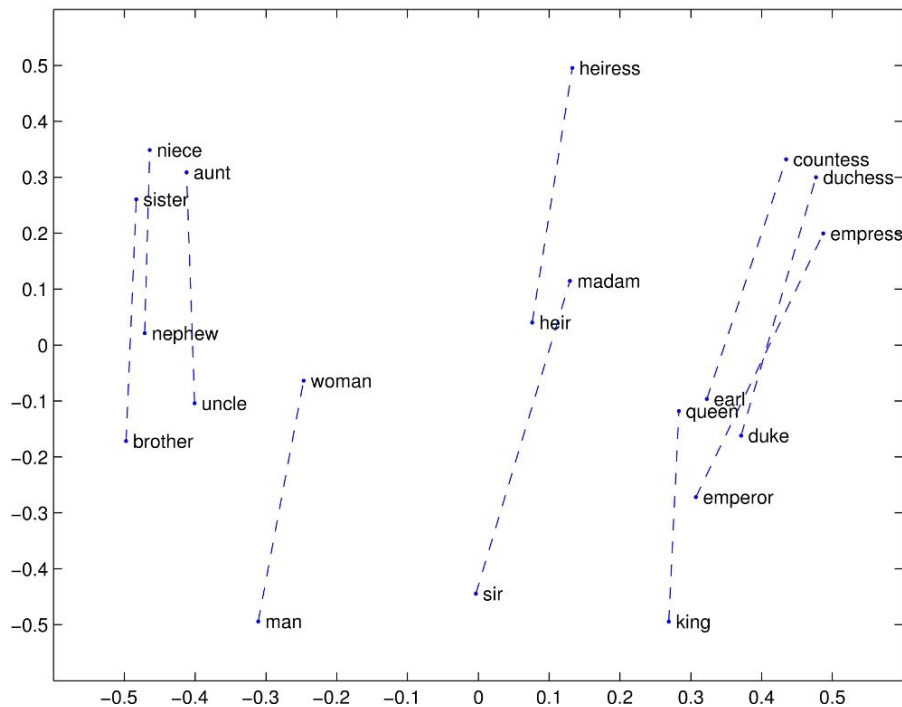
`chinese + river`

Ottengo un nuovo vettore, se cerco le parole più vicine trovo:

- `Yangtze River`
- `Yangtze`
- `Qiantang River`

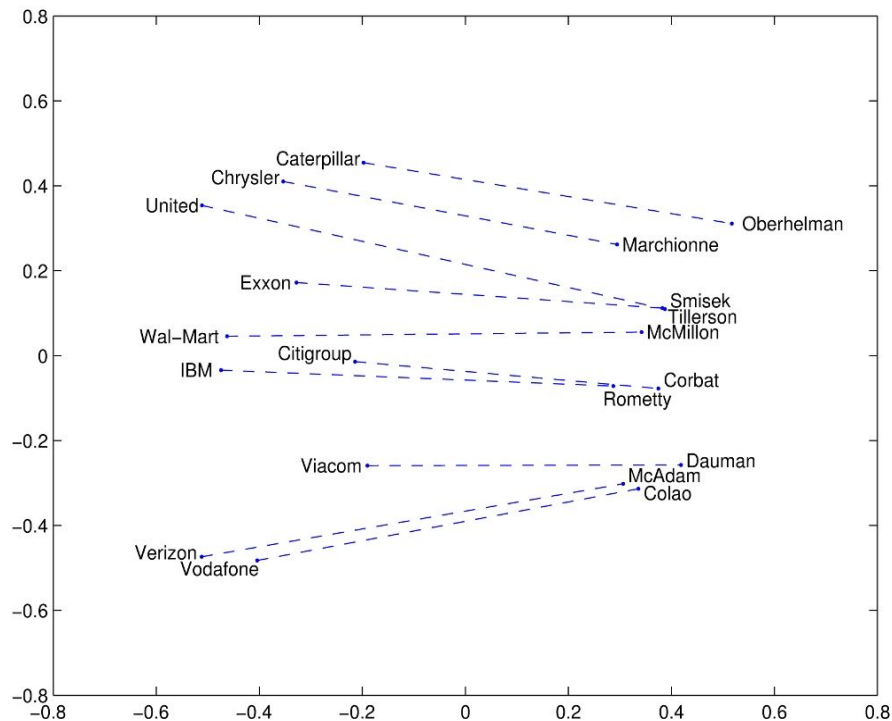
Maschio-Femmina

Le distanze tra maschile e femminile sono molto simili

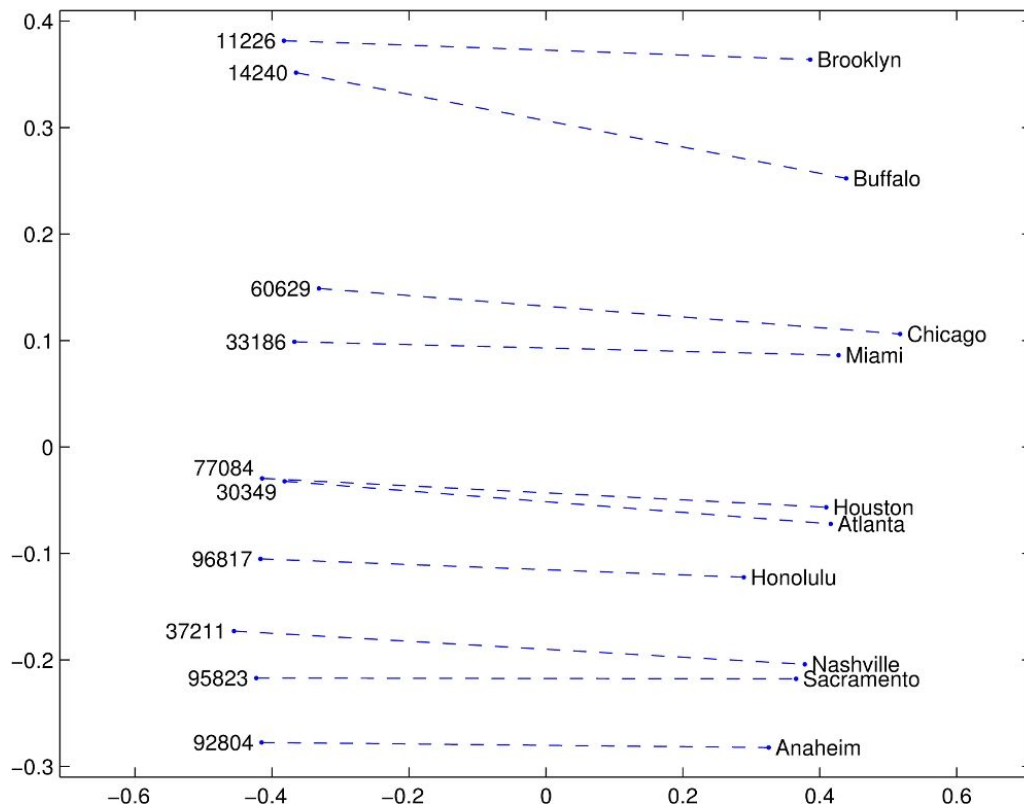


Company-CEO

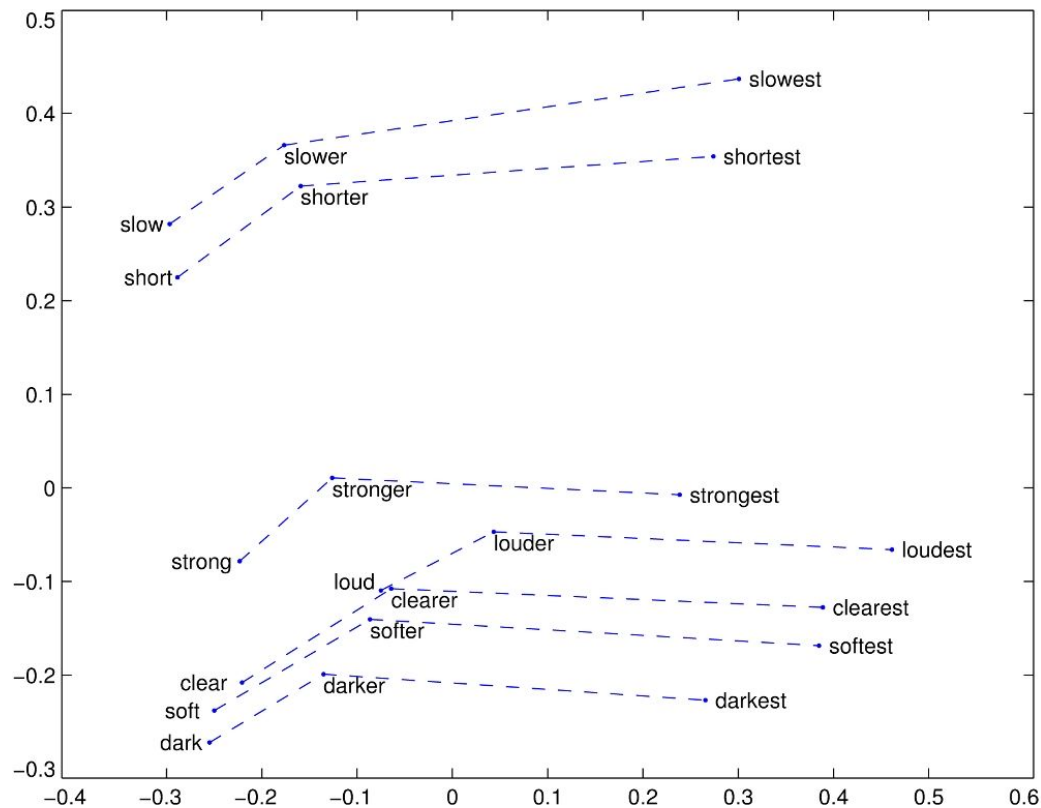
Anche distanze tra azienda e CEO sono molto simili



Città-ZIP Code



Comparativo-Superlativo



Risultato notevole

king - man + woman = queen

Word Embedding

Idea di base, giusto accennata...

- "Words that occur in the same contexts tend to have similar meanings"
(Harris 1954)
- Dobbiamo identificare il senso delle parole in base a quello che le circonda

The kid said he would grow up to be Superman

The child said he would grow up to be Superman

Skip-gram

- Ho una grande quantità di testo
- Costruisco un vocabolario
- Addestro una rete neurale:
 - Input: 1 parola
 - Output: le parole **più probabili** come parole vicine

Esempio skip-gram

Source Text

Training Samples

The quick brown fox jumps over the lazy dog. →

(the, quick)
(the, brown)

The quick brown fox jumps over the lazy dog. →

(quick, the)
(quick, brown)
(quick, fox)

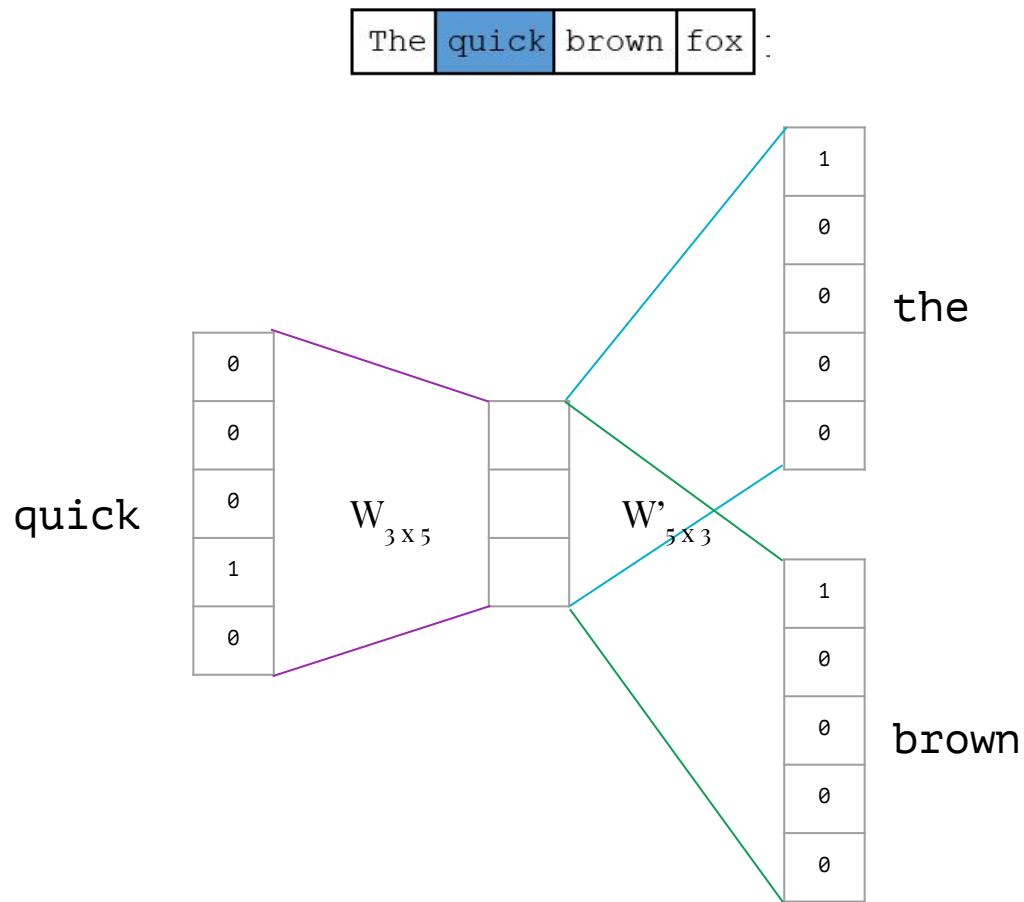
The quick brown fox jumps over the lazy dog. →

(brown, the)
(brown, quick)
(brown, fox)
(brown, jumps)

The quick brown fox jumps over the lazy dog. →

(fox, quick)
(fox, brown)
(fox, jumps)
(fox, over)

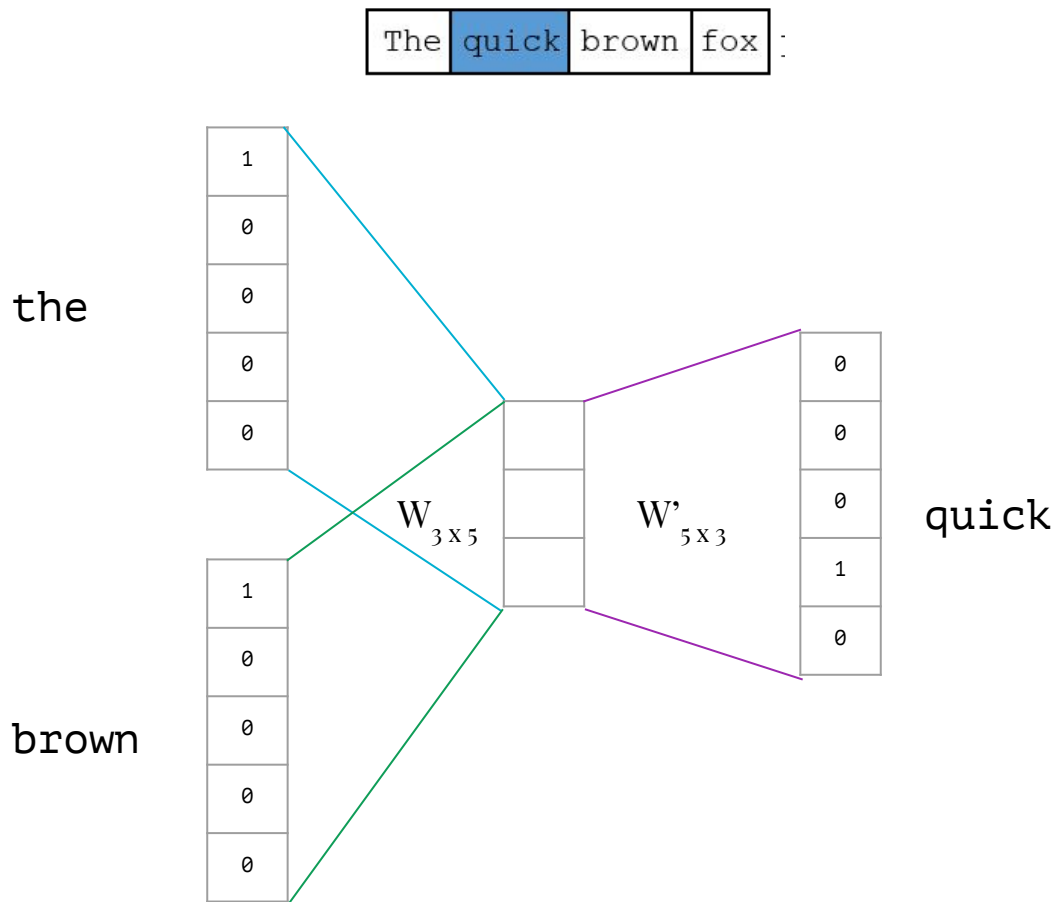
Skip-gram



CBOW (Continuous Bag Of Words)

- Ho una grande quantità di testo
- Costruisco un vocabolario
- Addestro una rete neurale:
 - Input: **più** parole (contesto)
 - Output: la parola **più probabile** per quel **contesto**

CBOW

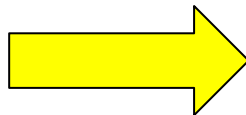


Traduzione automatica (cenni)

- Word embedding rappresentano parole in un nuovo spazio
- Trovo questo spazio partendo da documenti di una certa lingua
 - Embedding per **Italiano**
 - Embedding per **Tedesco**
 - Embedding per **Giapponese**

Embedding:

rana



0.16
0.63
...
0.01
0.57

Traduzione automatica (cenni)

Potrei fare l'inverso!

- Da un punto nell'embedding, torno alla parola
- Potrei farlo con un embedding in **inglese**, in **italiano**, in **tedesco**...

- Embedding per ogni lingua
 - Proietto su nuovo spazio
 - Da nuovo spazio torno indietro
- Traduzione
 - Proietto su spazio embedded lingua sorgente
 - Proiezione inversa su lingua target

Natural Language Generation

Non entriamo nel dettaglio, ma se posso sfruttare la possibilità di trovare il contesto di una parola...

- Prendo una parola a caso (o una sequenza di parole)
- Sfrutto la rete neurale per trovare la parola che **segue più probabile**
- (ho bisogno di una rete che abbia memoria)
- Aggiungo la nuova parola alla mia sequenza e ricomincio...

NLG

Ciao, mi chiamo

Ciao, mi chiamo **Andrea**

Ciao, mi chiamo Andrea **e**

Ciao, mi chiamo Andrea e **sono**

Ciao, mi chiamo Andrea e sono **un**

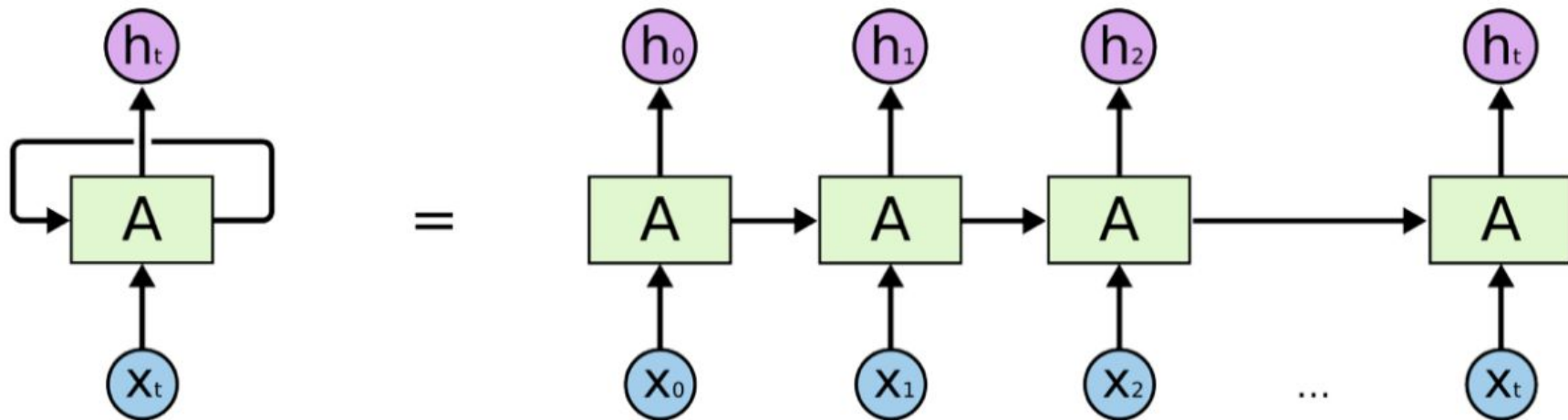
Ciao, mi chiamo Andrea e sono un **ricercatore**

Reti neurali ricorrenti

(ho bisogno di una rete che abbia memoria)

- Abbiamo usato reti che prendono **una** parola e ci danno un vettore
- Per generare testo siamo interessati all'**ordine** e al **contesto**
- Abbiamo bisogno di ricordare l'ordine in cui sono comparse le parole
- Ci serve una **memoria**

Reti neurali ricorrenti



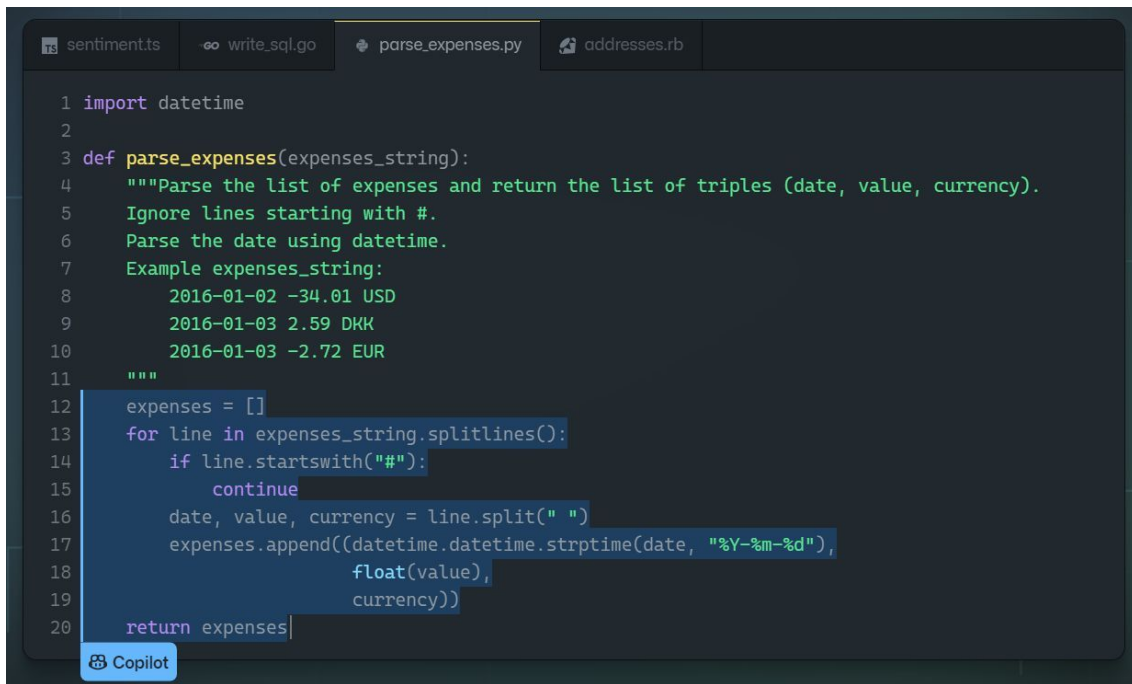
Impressionante

<https://6b.eleuther.ai/>



Generazione di codice

<https://copilot.github.com/>



The image shows a screenshot of a code editor with a dark theme. At the top, there are four tabs: 'sentiment.ts', 'write_sql.go', 'parse_expenses.py' (which is active), and 'addresses.rb'. The main editor area displays a Python function named 'parse_expenses'. The function takes 'expenses_string' as an argument and returns a list of expense records. Each record is a tuple containing a date (datetime object), a value (float), and a currency (string). The function includes a docstring with a description, instructions to ignore lines starting with '#', and an example of the input string. The code is as follows:

```
1 import datetime
2
3 def parse_expenses(expenses_string):
4     """Parse the list of expenses and return the list of triples (date, value, currency).
5     Ignore lines starting with #.
6     Parse the date using datetime.
7     Example expenses_string:
8         2016-01-02 -34.01 USD
9         2016-01-03 2.59 DKK
10        2016-01-03 -2.72 EUR
11    """
12    expenses = []
13    for line in expenses_string.splitlines():
14        if line.startswith("#"):
15            continue
16        date, value, currency = line.split(" ")
17        expenses.append((datetime.datetime.strptime(date, "%Y-%m-%d"),
18                        float(value),
19                        currency))
20    return expenses
```

At the bottom left of the editor, there is a blue button with the GitHub logo and the text 'Copilot'.