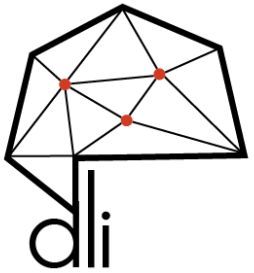
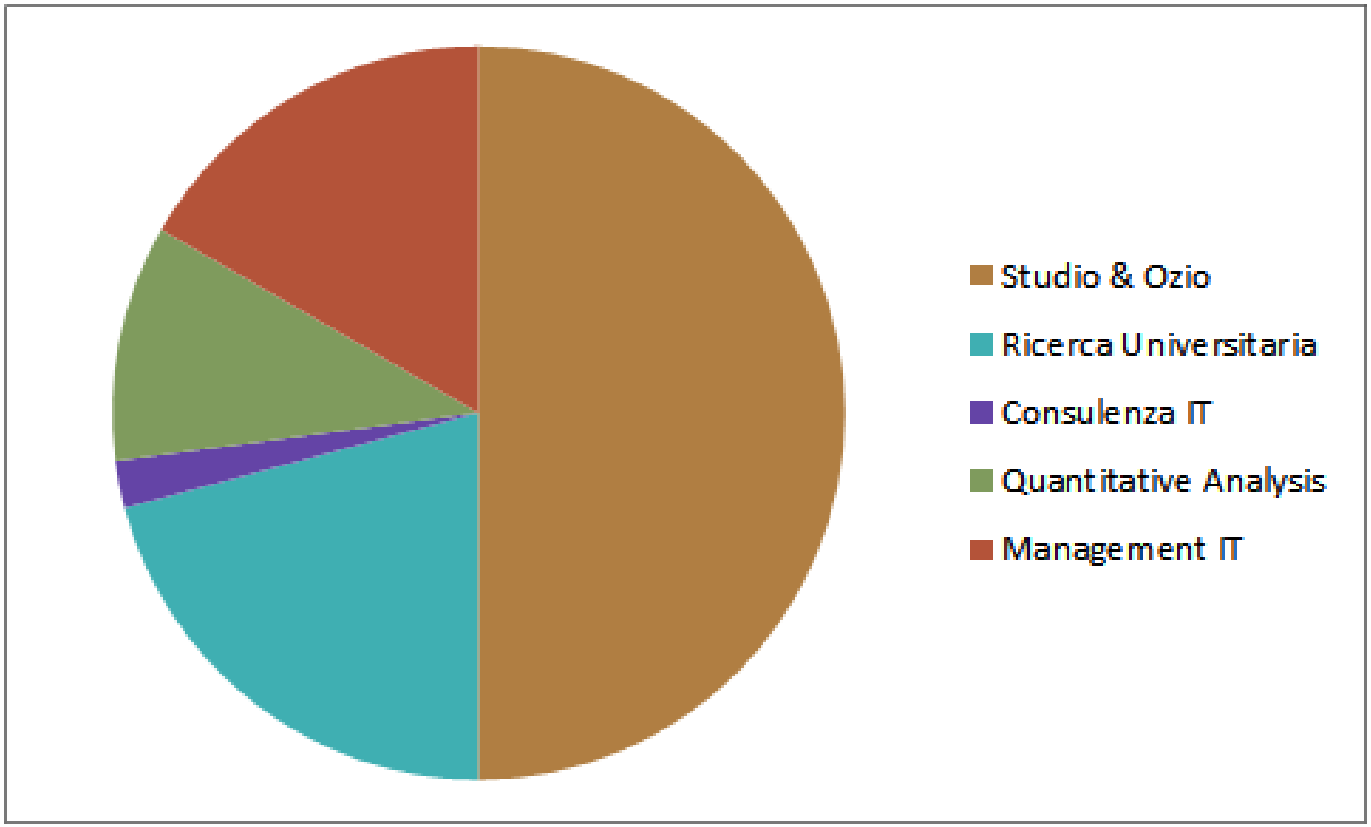


Processare il linguaggio naturale

Paolo Caressa

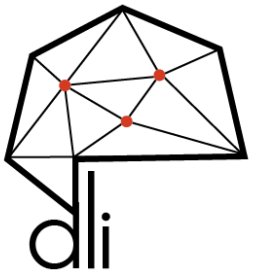
deeplearningitalia.com 20 maggio 2019

Lo speaker: una vita con... torta



http://www.caressa.it
https://www.linkedin.com/in/paolocaressa
@www_caressa_it

Agenda



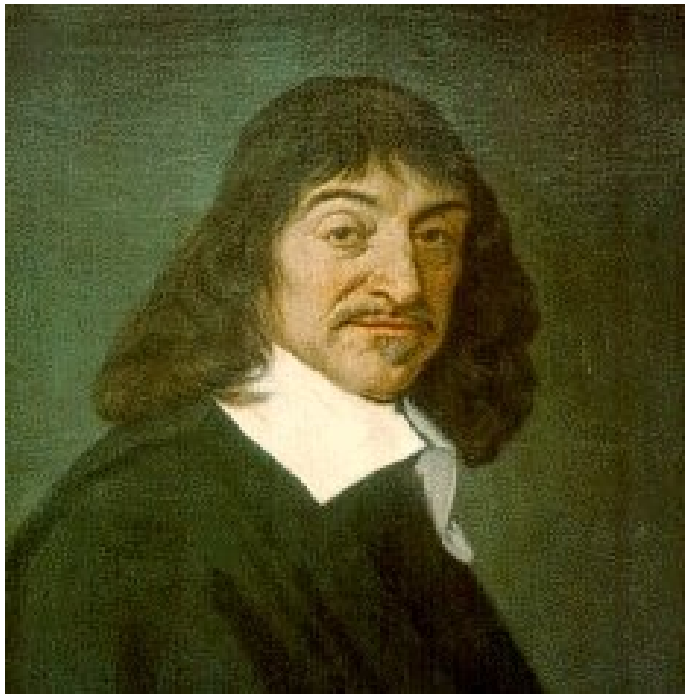
- **NLP: elaborazione automatica del linguaggio**
- Modelli statistici per la semantica: Markov
  - Demo time!
- Modelli probabilistici: Bayes
  - Demo time!
- Semantica distribuzionale
- Modelli vettoriali: word embedding
  - Demo time!
- Richiami sulle reti neurali
- Un esempio: Word2Vector
  - Demo time!

Il sogno di Leibniz...

L'unico modo di rendere corretti i nostri ragionamenti è di farli tangibili come quelli dei matematici, in modo da trovare gli errori semplicemente guardando, e quando ci siano dispute fra persone si possa semplicemente dire: calcoliamo senza indugio per vedere chi ha ragione.

Arte della scoperta (1685)

... pensando a Cartesio

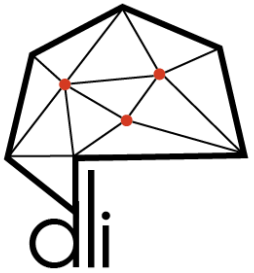


Il grande filosofo francese aveva nel XVII secolo "algebrizzato" la geometria, consentendo di sostituire ragionamenti e costruzioni geometriche con calcoli algebrici: Leibniz voleva fare lo stesso per qualsiasi tipo di ragionamento e deduzione, esprimibili nel linguaggio umano...

Un sogno che prende forma...

Il Word Embedding insegue, in un certo senso, il sogno di Leibniz usando proprio i metodi di Cartesio!!!

L'approccio formale



La teoria dei linguaggi formali, che si potrebbe chiamare anche "linguistica matematica", tenta di descrivere in modo generale cosa sia un linguaggio e come possa formalizzarsi matematicamente.

In particolare lo studio della sintassi e il suo legame con la teoria degli automi (gerarchie di Noam Chomsky) ha consentito di ingegnerizzare la costruzione dei "compilatori", cioè i programmi che traducono da un linguaggio di programmazione a un altro.

Ma le lingue umane non sono semplici e precise come quelle artificiali...

I limiti dell'approccio formale

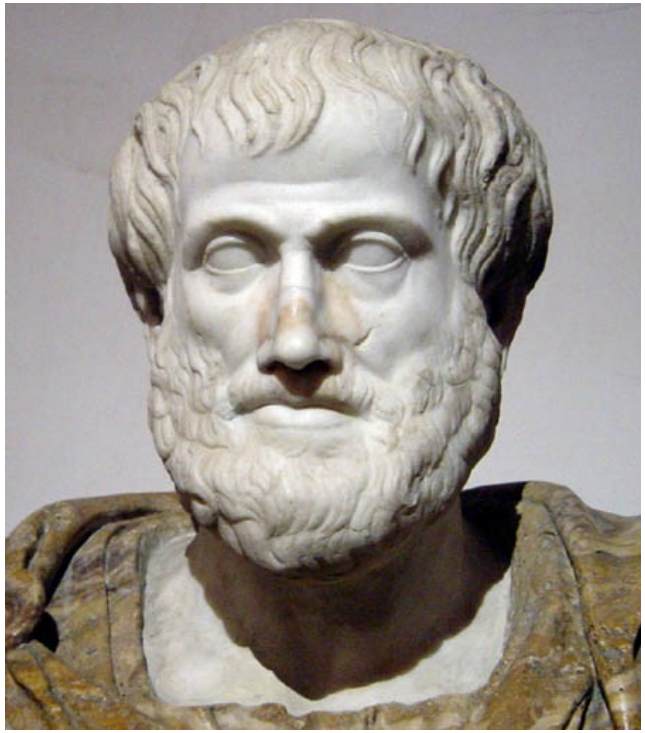
Il *sillogismo categorico,* codificato da Aristotele, è una formalizzazione di un frammento del linguaggio per svolgere deduzioni in modo automatico:

Socrate e Platone sono uomini

Gli uomini sono mortali

dunque

Socrate e Platone sono mortali



Indubitabile, no?

I limiti dell'approccio formale

O no? Guardate cosa si è inventato il matematico Giuseppe Peano:

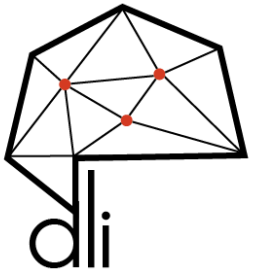
Pietro e Giovanni sono apostoli

Gli apostoli sono dodici

dunque

Pietro e Giovanni sono dodici

Le tre dimensioni del linguaggio

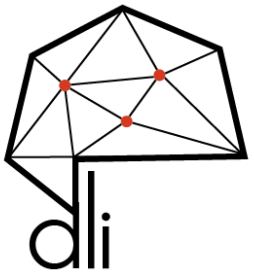


Le lingue umane sono caratterizzate almeno da (C.W. Morris, 1938):

- *Sintassi*: la struttura delle sequenze di simboli che compongono parole, frasi, etc. (analisi grammaticale).
- *Semantica*: i significati possibili di un testo (analisi logica).
- *Pragmatica*: il contesto (= con testo) di riferimento del significato (comprensione del testo).

Il significato del testo è l'oggetto centrale che si proietta in ciascuna di queste tre dimensioni concettuali.

Le tre dimensioni del linguaggio



Le lingue umane sono cara
Morris, 1938):

- *Sintassi*: la struttura d ne compongono paro frasi, (a grammaticale).
- *Semantica*: i significati possibili di un testo (analisi logica).
- *Pragmatica*: il contesto (= con testo) di riferimento del significato (comprensione del testo).

A noi interessa la semantica!!!

Il significato del testo è l'oggetto centrale che si proietta in ciascuna di queste tre dimensioni concettuali.

Il ruolo del contesto...

"Ho visto mangiare un cane"

Il ruolo del contesto...

"Ho visto mangiare un cane"

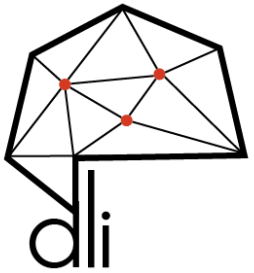
Il ruolo del contesto...

"Ho visto mangiare un cane"

Il ruolo del contesto...

"Ho visto mangiare un cane"

La parola dei Maestri...

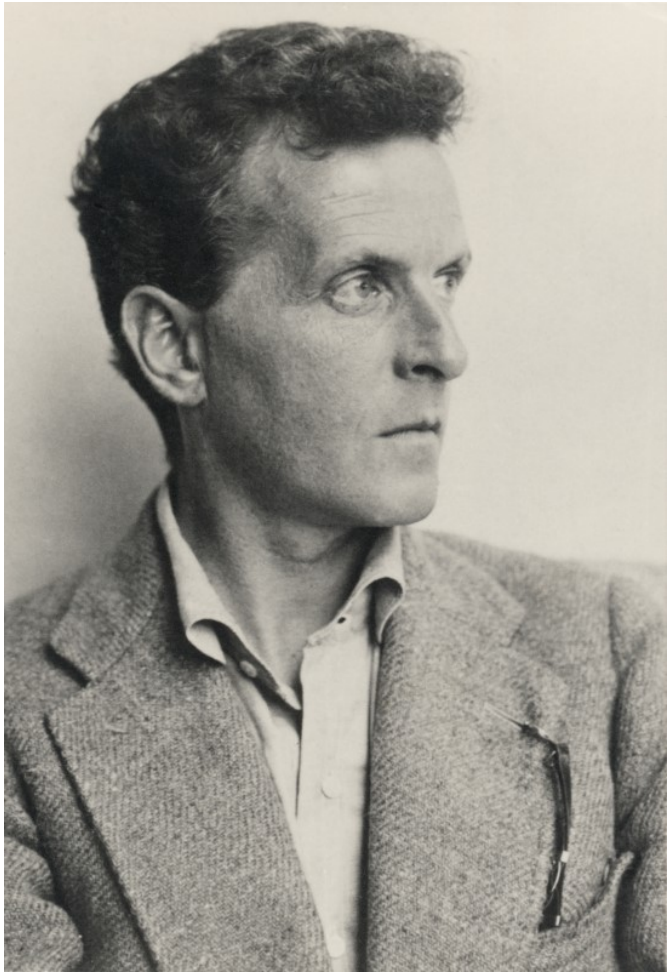


Da millenni, lo studio del linguaggio è stato oggetto di ricerche da parte di logici, filosofi e matematici (o varie combinazioni dei tre) e queste ricerche hanno anche avuto un diretto influsso sulla scienza dei calcolatori.

In molti, anche in conflitto fra loro, si sono interrogati su cosa sia il linguaggio e come sia possibile analizzarlo razionalmente.

Alcune di queste idee sono filtrate nell'informatica e hanno costituito la base per nuove categorie di algoritmi.

Ludwig Wittgenstein



Nella maggior parte dei casi dell'impiego della parola "significato", sebbene non in tutti, questa parola può essere spiegata come segue: il significato di una parola è l'uso che se ne fa nel linguaggio.
(Postumo)

Alfred Tarski

C'è chi parla di comprendere il vero e proprio significato di una nozione, qualcosa di indipendente dall'uso effettivo e indipendente dalle proposte normative, qualcosa come l'idea platonica dietro la nozione. Quest'ultimo approccio mi è così alieno ed estraneo che lo ignorerò in quanto non posso dire nulla di intelligente in proposito.

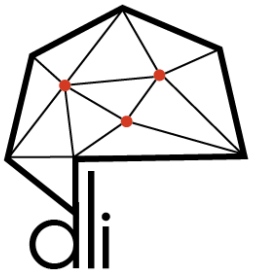
(1966)

Richard Montague

Secondo me non c'è alcuna differenza teorica rilevante fra lingue naturali e le lingue artificiali dei logici; infatti considero possibile racchiudere la sintassi e la semantica di entrambi questi tipi di linguaggi in una sola e naturale teoria matematicamente precisa.

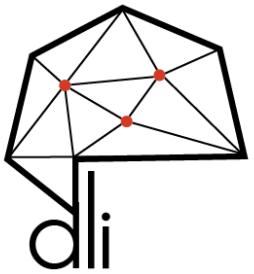
(1970)

Agenda



- NLP: elaborazione automatica del linguaggio
- **Modelli statistici per la semantica: Markov**
  - Demo time!
- Modelli probabilistici: Bayes
  - Demo time!
- Semantica distribuzionale
- Modelli vettoriali: word embedding
  - Demo time!
- Richiami sulle reti neurali
- Un esempio: Word2Vector
  - Demo time!

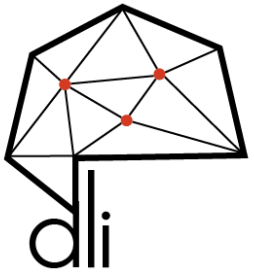
Mettiamo ordine fra le parole



Una prima idea per far emergere la "semantica" di un testo, è notare che le parole in una frase sono disposte in un certo *ordine*, dettato dalle regole grammaticali e da ciò che si vuole esprimere.

```
Paolo cuoce la pasta e poi la mangia
Paolo mangia la pasta e poi la cuoce
Poi la pasta la cuoce Paolo e mangia
La pasta la mangia e poi cuoce Paolo
Cuoce e la la mangia Paolo pasta poi
```

La grammatica non basta

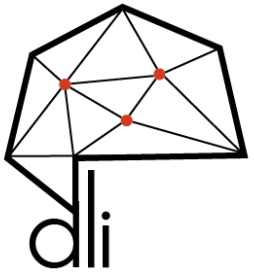


In un testo, dopo una parola non ne può seguire una qualsiasi altra, ma solo quelle che *ha senso* porre dopo la parola data, sia in termini grammaticali che semantici. Per esempio, in italiano prima del verbo va solitamente un soggetto. Tuttavia la frase

```
Il gatto abbaia
```

è grammaticalmente corretta ma priva di senso: il verbo segue il soggetto, ma non è il verbo giusto associato al soggetto giusto...

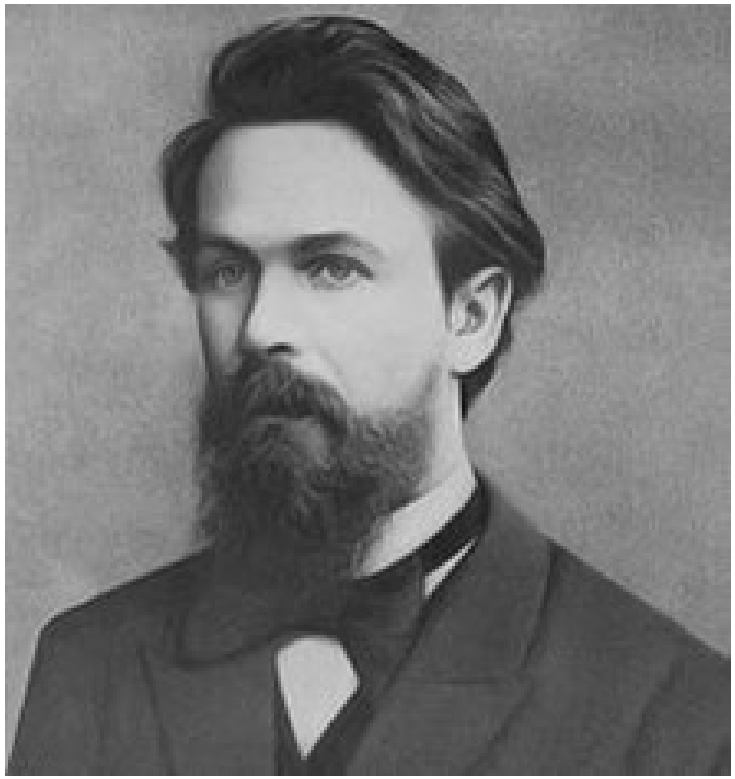
Perché l'approccio statistico?



Seguendo l'indicazione dei Maestri, per cui il significato del linguaggio sta nell'uso che se ne fa, potremmo partire non dalle regole astratte della grammatica di una lingua ma dai testi scritti in quella lingua e far emergere da essi le regole (ma anche le eccezioni) grammaticali.

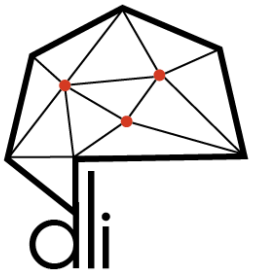
Un modo per farlo è analizzare tantissimi testi e vedere le relazioni di prossimità delle parole più probabili, cioè condurre una *analisi statistica* dei testi.

Andrej Andreevič Markov



La prima applicazione (1913) delle sue "catene di Markov" è stata il conteggio delle occorrenze di lettere nell'*Eugeny Onegin* di Puškin, per capire quando una vocale segue un'altra vocale (12,8% dei casi) o una consonante (66,3% dei casi).

Un modello markoviano



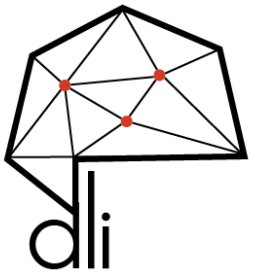
Si può applicare l'idea di Markov non alle singole lettere, ma alle parole. Denotiamo con P_{xy} la probabilità che la parola x segua la parola y. Per esempio, se $P_{xy} = 0$ vuol dire che la parola x non compare mai dopo la parola y.

Avendo a disposizione un corpus di testi in una lingua, possiamo stimare P:

$$P_{xy} = \frac{numero\ di\ volte\ che\ x\ segue\ y}{numero\ di\ volte\ che\ x\ segue\ una\ qualsiasi\ parola}$$

Si tratta di un modo "frequentista" di stimare le probabilità.

Un modello markoviano

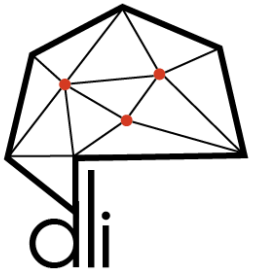


Se per tutti i documenti di un corpus calcoliamo P_{xy} per tutte le coppie di parole che occorrono una di seguito all'altra, fissata y, le P_{xy} danno una distribuzione di probabilità che possiamo usare per estrarre a caso una parola x fra quelle che *possono* seguire y.

Poi estraiamo una parola w fra quelle che possono seguire x, e così via, in modo da generare sequenze di parole che sono "possibili" nel linguaggio dato in base all'uso che se ne fa nei testi disponibili.

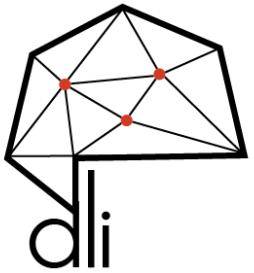
Usiamo questa tecnica per un esperimento un po' surreale...

Agenda



- NLP: elaborazione automatica del linguaggio
- Approccio statistico alla semantica
  - **Demo time!**
- Modelli probabilistici: Bayes
  - Demo time!
- Semantica distribuzionale
- Modelli vettoriali: word embedding
  - Demo time!
- Richiami sulle reti neurali
- Un esempio: Word2Vector
  - Demo time!

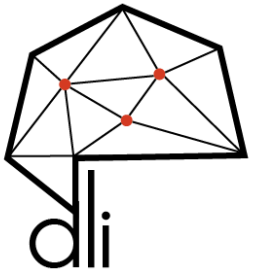
Un semplice modello markoviano



Ecco un semplicissimo programma basato sul modello di Markov che risponda a domande come avrebbe fatto Antonio Gramsci: per farlo

- ho scaricato i tre tomi degli "Scritti politici" di Gramsci da Internet (https://www.liberliber.it/online/autori/autori-g/antonio-gramsci/)
- ho scritto una libreria per calcolare le probabilità P_{xy} per questo corpus (la parte più divertente del coding).
- ho scritto un programma che, usando la libreria, chiede di inserire una domanda e, partendo dall'ultima parola di quest'ultima, generi una risposta (5 parole per ogni parola data, per consentire frasi parzialmente sensate).

Un semplice modello markoviano



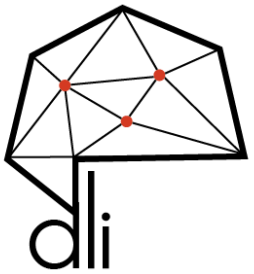
Ovviamente il risultato non ha molto senso: si chiama...

Conversazione con un pazzo che si crede Gramsci

Potete scaricare uno zip col programma Python, la libreria e i dati necessari da questa URL:
http://www.caressa.it/gramsci.zip

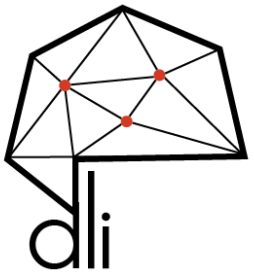
(cambiando il contenuto del file "corpus.txt", cancellando il file .pkl e rilanciando il programma potrete farlo conversare alla maniera di chi volete voi...)

Agenda



- NLP: elaborazione automatica del linguaggio
- Approccio statistico alla semantica
  - Demo time!
- **Modelli probabilistici: Bayes**
  - Demo time!
- Semantica distribuzionale
- Modelli vettoriali: word embedding
  - Demo time!
- Richiami sulle reti neurali
- Un esempio: Word2Vector
  - Demo time!

Probabilità condizionate



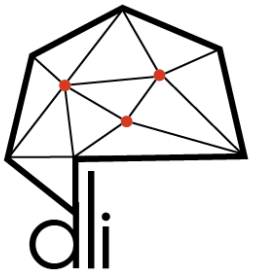
Torniamo per un momento ai valori P_{xy} che forniscono le probabilità che una parola x segua una parola y.

Queste probabilità si possono pensare anche come alle probabilità che la parola x appaia nel testo, avendo l'informazione che la parola y è appena apparsa: sono quindi quelle che i probabilisti chiamano *probabilità condizionate* P(y|x).

Sorge quindi l'idea di utilizzare una tecnica deduttiva fondamentale in moltissimi campi applicativi anche per studiare la struttura statistica di un linguaggio sulla base di testi scritti per mezzo di esso: il *teorema di Bayes*.

Thomas Bayes (1702-1761)

Modelli bayesiani

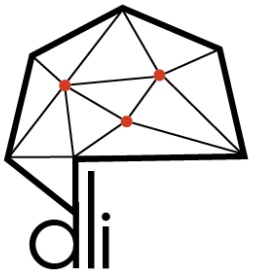


Ricordiamo brevemente come si applica il teorema di Bayes: supponiamo di avere un insieme di ipotesi h_1, ..., h_2 relative a dei dati (o osservazioni) d.

Vogliamo scegliere l'ipotesi più probabile dato d, quindi scegliere l'h_i per cui è massima la probabilità $P(h_i|d)$.

Il problema è che tipicamente questa "probabilità a posteriori" non è immediatamente nota: si pensi all'esempio di una malattia (h) da dedurre in base ai sintomi (d); solitamente sappiamo stimare la probabilità di avere certi sintomi data la presenza della malattia, cioè $P(d|h_i)$.

Il teorema di Bayes

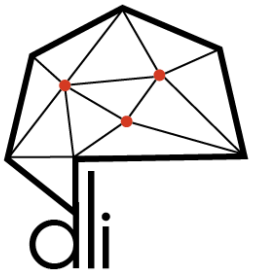


Il teorema di Bayes consente di calcolare la probabilità a posteriori di h_i date le sue probabilità a priori:

$$\mathrm{P}(h_i|d) = \frac{\mathrm{P}(d|h_i)\mathrm{P}(h_i)}{\mathrm{P}(d)}$$

Non sorprende che P($h_i|d$) sia proporzionale a P(h_i) ed a P($d|h_i$). Inoltre, più è elevata la probabilità a priori P(d), che quindi d sia osservata indipendentemente da h_i, e più decresce la probabilità a posteriori P($h_i|d$).

Scegliere l'ipotesi migliore

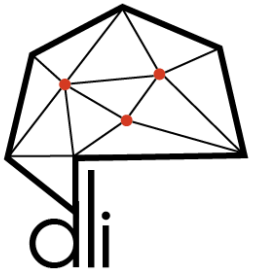


A questo punto, possiamo calcolare le probabilità a posteriori per ciascuna ipotesi e scegliere quella che le massimizza:

$$\begin{aligned} h_{best} &= \arg\max_i \mathrm{P}(h_i|d) \\ &= \arg\max_i \mathrm{P}(d|h_i)\mathrm{P}(h_i)/\mathrm{P}(d) \\ &= \arg\max_i \mathrm{P}(d|h_i)\mathrm{P}(h_i) \end{aligned}$$

Se le h_i sono equiprobabili abbiamo la *Maximum Likelihood* (in quanto possiamo non considerare P(h_i) nel calcolare l'indice *i* per il quale l'argomento è massimo)..

Classificazione di documenti

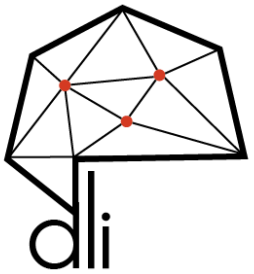


Una classica applicazione dei modelli bayesiani alla elaborazione digitale dei testi ne concerne la classificazione.

Vogliamo cioè assegnare dei "tag" a documenti di testo in modo che documenti che hanno un contenuto analogo condividano lo stesso tag.

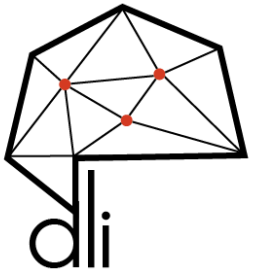
Per mostrare in concreto questo esempio ben noto, l'abbiamo implementato formulando le seguenti ipotesi:

Ipotesi



- Consideriamo un corpus di documenti dai quali estraiamo tutte le parole e le poniamo in un "vocabolario" *V*.
- Un singolo documento *d* viene rappresentato in questo modo: se *n* è il numero delle sue parole (anche ripetute), il documento si identifica con una funzione $d:\{1,2,...,n\} \rightarrow V$ in modo che associ al valore i la parola in quella posizione.
- Un documento possiede uno ed un solo tag: diciamo che il documento *d* appartiene alla classe *h* se il suo tag è *h*.

Ipotesi "naive Bayes"



Nell'applicare l'algoritmo di Bayes

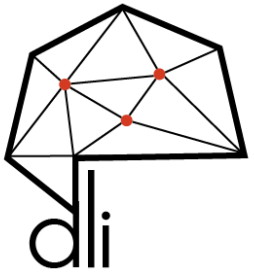
$$h_{best} = \arg\max_i \mathrm{P}(h_i|d)$$

d è nel nostro caso una funzione, determinata dall'insieme {*d*(1),...,*d*(n)} dei suoi valori. A priori non è detto che questi valori siano indipendenti (per esempio a un articolo seguirà un nome): l'ipotesi "ingenua" è che lo siano e quindi

$$h_{best} = \arg\max_i \left[\mathrm{P}(d(1)|h_i) \cdots \mathrm{P}(d(n)|h_i)\right]\mathrm{P}(h_i)$$

(a e b sono indipendenti se P(a|b) = P(a) e P(b|a) = P(b), da cui segue che P(a,b) = P(a)P(b).

Ipotesi "naive Bayes"

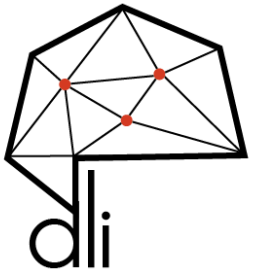


L’ipotesi "ingenua" è quindi in questo caso che le posizioni delle parole non influenzano le loro probabilità di comparire in una data posizione.

Questo è chiaramente falso ed è la negazione dell’ipotesi alla base del modello markoviano.

Tuttavia questa palese falsità semplifica computazionalmente l’algoritmo in modo formidabile.

Ipotesi "naive Bayes"

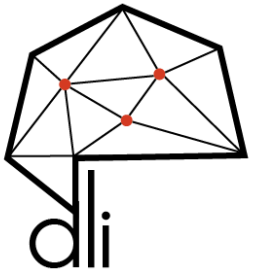


Nel caso specifico dei documenti di testo, questa ipotesi si può formalizzare come, per ogni *h*, *k*

$$\mathrm{P}(d(h) = w|h_i) = \mathrm{P}(d(k) = w|h_i)$$

Cioè, data una classe, le parole di un documento hanno la stessa probabilità a posteriori di comparire indipendentemente dalla posizione che occupano nel documento.

Ipotesi "naive Bayes"

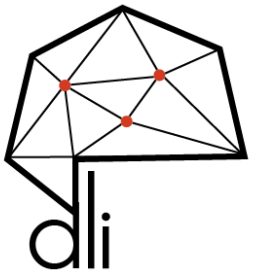


In questo modo possiamo stimare le probabilità $\mathrm{P}(d(h) = w|h_i)$ semplicemente come $\mathrm{P}(w|h_i)$, cioè calcolando la frequenza di *w* in tutti i documenti della classe h_i e dividendola per tutte le parole che occorrono in quella classe.

Un miglioramento di questa stima è

$$P(w|h_i) = \frac{\#\{\text{occorrenze di } w \text{ in } h_i\} + 1}{\#\{\text{occorrenze qualsiasi in } h_i\} + \#\{\text{parole}\}}$$

Utilizzare un "training set"

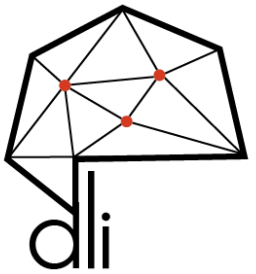


L’algoritmo "naive Bayes" richiede un "addestramento" su un insieme di documenti la cui classificazione sia nota.

Si tratta cioè di un algoritmo di *machine learning supervisionato*, che per poter esprimere un parere su dati di classificazione ignota deve aver elaborato numerosi esempi classificati.

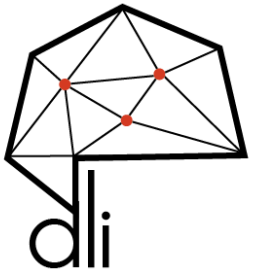
Precisamente, l’algoritmo ha necessità di un numero sufficiente di documenti classificati per poter stimare le probabilità a posteriori.

Agenda



- NLP: elaborazione automatica del linguaggio
- Approccio statistico alla semantica
  - Demo time!
- Modelli probabilistici: Bayes
  - **Demo time!**
- Semantica distribuzionale
- Modelli vettoriali: word embedding
  - Demo time!
- Richiami sulle reti neurali
- Un esempio: Word2Vector
  - Demo time!

Un esempio classico

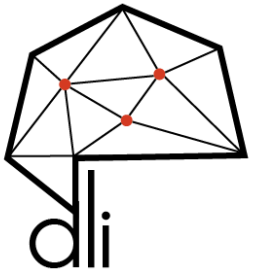


Nel nostro caso abbiamo usato il classico repository "20 newsgroups", che contiene 20.000 documenti di testo tratti da news degli anni ‘90.

Ciascuna si riferisce a delle categorie particolari di interesse, 20 in tutto, e ciascuna contiene 1.000 documenti.

Scegliendo come training set 2/3 del corpus, vale a dire i primi 666 documenti, e lasciando i restanti 334 come testing set, abbiamo ottenuto una percentuale di 80% di classificazioni corrette su esempi che il programma non aveva incontrato prima.

Un esempio classico



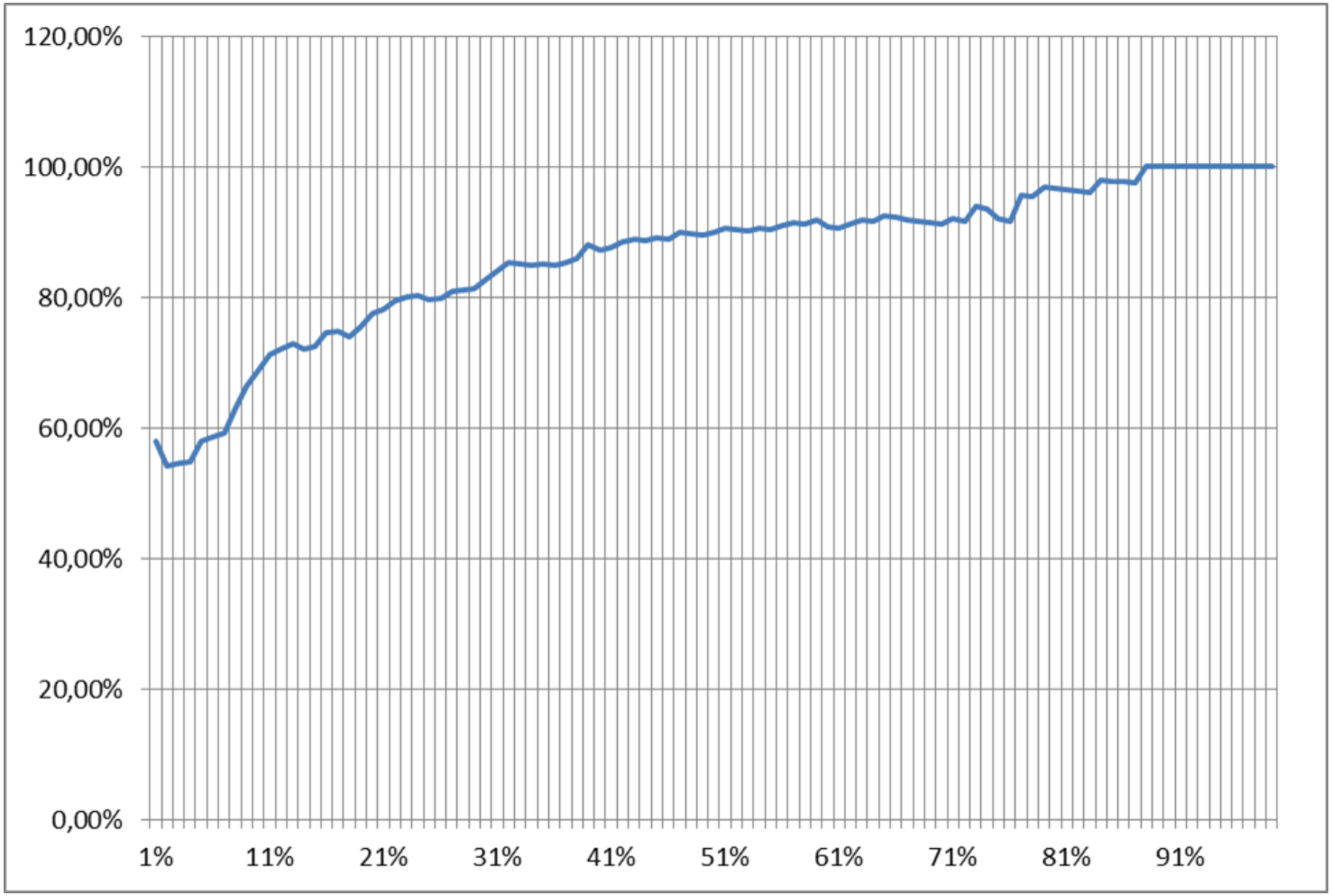
Limitandosi a sole tre classi e 100 file in totale per ogni classe si può mostrare facilmente come varia la percentuale di risposte esatte E rispetto alla percentuale di documenti utilizzati per il training T.

Si verifica che

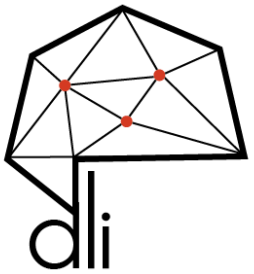
- se T > 25% allora E > 80%
- se T > 50% allora E > 90%
- se T > 88% allora E = 100%

Nel grafico seguente, sulle ascisse i valori di T, sulle ordinate le risposte esatte corrispondenti.

Un esempio classico



Ecco il programma

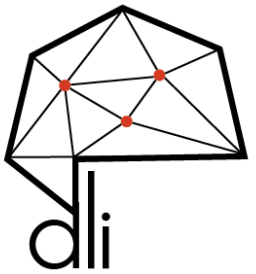


Il programma va messo in una cartella che contiene una cartella con i file del dataset, ciascuno in una sottocartella rappresentante la classe corrispondente.

nbayes_test.py svolge il test (chiede in input la percentuale di documenti in una classe usati per il training)
nbayes.py la libreria che calcola naive Bayes
20 Newsgroup dataset su cui provare il programma (che deve stare nella stessa cartella in cui si decomprime il file)

.

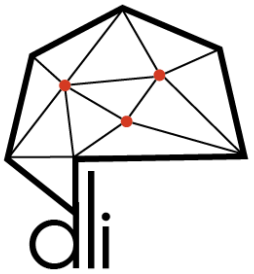
Esempio di utilizzo



Risultato dell'esecuzione del programma nel caso di 3 classi e 100 documenti per classe:

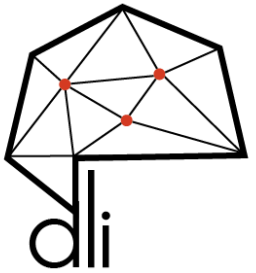
```
Carica nel vocabolario tutte le parole di tutti i documenti
alt.atheism
comp.graphics
rec.sport.hockey
Percentuale di documenti nel training set: 66

Training set: 66 %
Classe alt.atheism contiene 31075 parole
Classe comp.graphics contiene 14642 parole
Classe rec.sport.hockey contiene 22003 parole

Fase di test: classifica i documenti che non sono nel training set.
Classe alt.atheism 82.3529411764706% di risposte esatte
Classe comp.graphics 100.0% di risposte esatte
Classe rec.sport.hockey 94.11764705882354% di risposte esatte

Risultato totale della classificazione: 92.15686274509804 % di risposte esatte
Premere un tasto per terminare...
```

Agenda



- NLP: elaborazione automatica del linguaggio
- Approccio statistico alla semantica
  - Demo time!
- Modelli probabilistici: Bayes
  - Demo time!
- **Semantica distribuzionale**
- Modelli vettoriali: word embedding
  - Demo time!
- Richiami sulle reti neurali
- Un esempio: Word2Vector
  - Demo time!

Una generalizzazione dei modelli markoviani



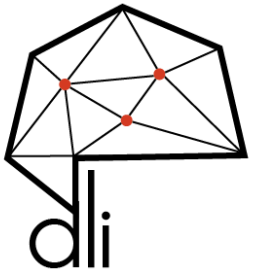
L'idea del *toy model* precedente è di modellizzare statisticamente le relazioni fra coppie di parole consecutive in un testo.

Più in generale potremmo, dato un testo le cui parole, nell'ordine e anche con ripetizioni, sono

$$p_1 \; p_2 \; p_3 \; p_4 \; p_5 \; \ldots \; p_N$$

scorrere tutte le parole (per i = 1, 2, ..., N) e considerare per ciascuna parola le k che la precedono e le k che la seguono:

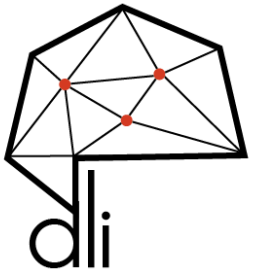
Una generalizzazione dei modelli markoviani



Esempio (con k = 2):

"Quel ramo del lago di Como, che volge a mezzogiorno, tra due **catene** non interrotte di monti, tutto a seni e a golfi, a seconda dello sporgere e del rientrare di quelli, vien, quasi a un tratto, a ristringersi, e a prender corso e figura di fiume, tra un promontorio a destra, e un'ampia costiera dall'altra parte"

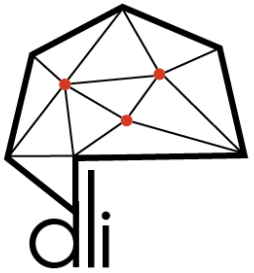
Una generalizzazione dei modelli markoviani



Esempio (con k = 2):

"Quel ramo del lago di Como, che volge a mezzogiorno, tra due catene **non** interrotte di monti, tutto a seni e a golfi, a seconda dello sporgere e del rientrare di quelli, vien, quasi a un tratto, a ristringersi, e a prender corso e figura di fiume, tra un promontorio a destra, e un'ampia costiera dall'altra parte"

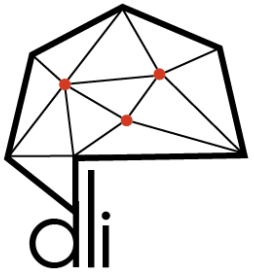
Una generalizzazione dei modelli markoviani



Esempio (con k = 2):

"Quel ramo del lago di Como, che volge a mezzogiorno, tra due catene non **interrotte** di monti, tutto a seni e a golfi, a seconda dello sporgere e del rientrare di quelli, vien, quasi a un tratto, a ristringersi, e a prender corso e figura di fiume, tra un promontorio a destra, e un'ampia costiera dall'altra parte"

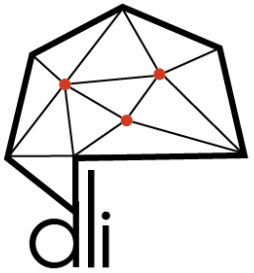
Una generalizzazione dei modelli markoviani



Esempio (con k = 2):

"Quel ramo del lago di Como, che volge a mezzogiorno, tra due catene non interrotte **di** monti, tutto a seni e a golfi, a seconda dello sporgere e del rientrare di quelli, vien, quasi a un tratto, a ristringersi, e a prender corso e figura di fiume, tra un promontorio a destra, e un'ampia costiera dall'altra parte"

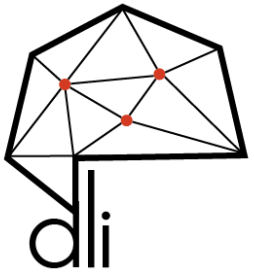
Una generalizzazione dei modelli markoviani



Esempio (con k = 2):

"Quel ramo del lago di Como, che volge a mezzogiorno, tra due catene non interrotte di **monti**, tutto a seni e a golfi, a seconda dello sporgere e del rientrare di quelli, vien, quasi a un tratto, a ristringersi, e a prender corso e figura di fiume, tra un promontorio a destra, e un'ampia costiera dall'altra parte"

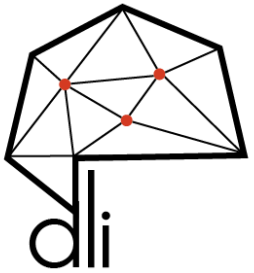
Una generalizzazione dei modelli markoviani



Esempio (con k = 2):

"Quel ramo del lago di Como, che volge a mezzogiorno, tra due catene non interrotte di monti, **tutto** a seni e a golfi, a seconda dello sporgere e del rientrare di quelli, vien, quasi a un tratto, a ristringersi, e a prender corso e figura di fiume, tra un promontorio a destra, e un'ampia costiera dall'altra parte"

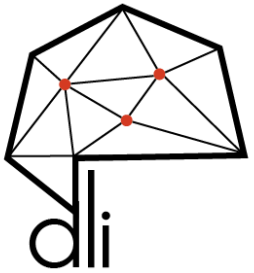
Una generalizzazione dei modelli markoviani



Esempio (con k = 2):

"Quel ramo del lago di Como, che volge a mezzogiorno, tra due catene non interrotte di monti, tutto **a** seni e a golfi, a seconda dello sporgere e del rientrare di quelli, vien, quasi a un tratto, a ristringersi, e a prender corso e figura di fiume, tra un promontorio a destra, e un'ampia costiera dall'altra parte"

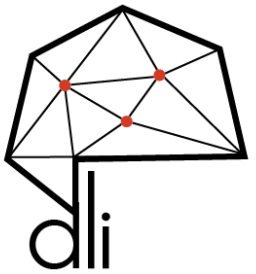
Una generalizzazione dei modelli markoviani



Esempio (con k = 2):

"Quel ramo del lago di Como, che volge a mezzogiorno, tra due catene non interrotte di monti, tutto a **seni** e a golfi, a seconda dello sporgere e del rientrare di quelli, vien, quasi a un tratto, a ristringersi, e a prender corso e figura di fiume, tra un promontorio a destra, e un'ampia costiera dall'altra parte"

Il contesto di una parola...



L'algoritmo appena "abbozzato" può essere usato per determinare un modello semantico delle parole di un corpus, considerando delle "finestre mobili" di parole costruite attorno alle parole da analizzare.

L'idea è che il contesto di una parola, che ne può determinare il significato, sono le parole che occorrono nelle "vicinanze" di essa in un testo (per esempio fra le 5 parole che la precedono e le 5 che la seguono).

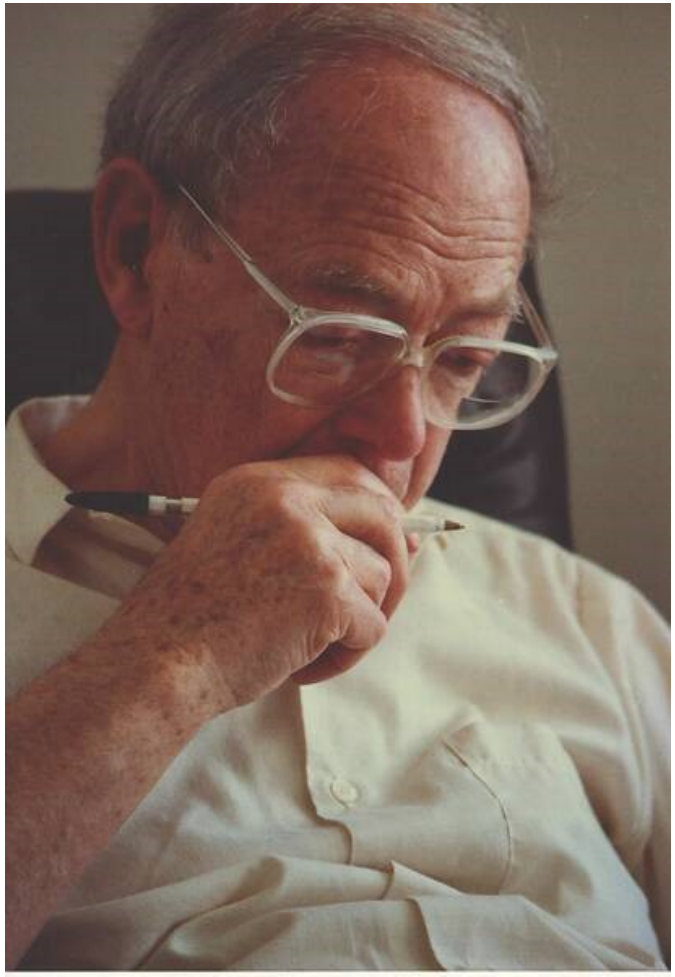
Come sempre accade, questa idea non è recente e non l'hanno avuta gli informatici...

Fondamenti teorici dell'analisi semantica distribuita

Una parola è caratterizzata dalla compagnia in cui si trova.

John Rupert Firth

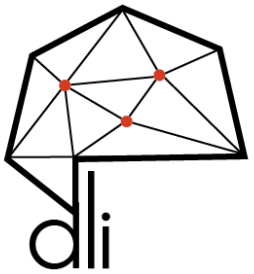
Fondamenti teorici dell'analisi semantica distribuita



La correlazione fra linguaggio e significato è molto più grande quando consideriamo un discorso connesso. Nella misura in cui la struttura distribuzionale si può scoprire in un discorso, essa è in qualche modo correlata con la sostanza di quanto viene detto; questo è evidente in special modo nei discorsi scientifici.

Zellig Harris

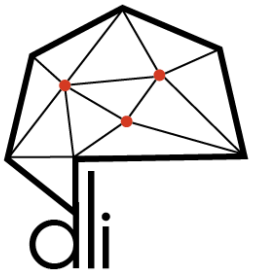
Analisi semantica distribuita



Si tratta di una tecnica che consiste nell'analizzare un corpus di documenti estraendone le parole e i loro contesti, usandoli per formare una tabella che poi viene ridotta nelle sue dimensioni usando algoritmi di algebra lineare (SVD).

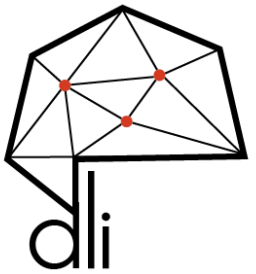
Le righe di questa tabella ridotta costituiscono quindi dei "vettori numerici" associati alle parole, che ne codificano la semantica, in modo che parole con significato analogo tendono a essere rappresentate da vettori "vicini" (fra poco richiameremo questi concetti).

Applicazioni dell'analisi semantica distribuita



- Rispondere a query sul corpus di documenti che siano più sofisticate della mera ricerca delle parole.
- Trovare gruppi di parole simili, cioè data una parola determinarne quelle che hanno un significato analogo.
- Estensione da singole parole a gruppi di parole per individuare la semantica di frasi.
- Individuare l'argomento di cui parla un documento (anche sentiment analysis).

Agenda

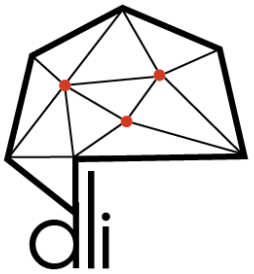


- NLP: elaborazione automatica del linguaggio
- Approccio statistico alla semantica
  - Demo time!
- Modelli probabilistici: Bayes
  - Demo time!
- Semantica distribuzionale
- **Modelli vettoriali: word embedding**
  - Demo time!
- Richiami sulle reti neurali
- Un esempio: Word2Vector
  - Demo time!

Facile come l'algebra...

Joseph Louis Lagrange ebbe a dire del suo amico Antoine Lavoisier che il suo grande merito era stato di rendere la chimica *facile come l'algebra*.

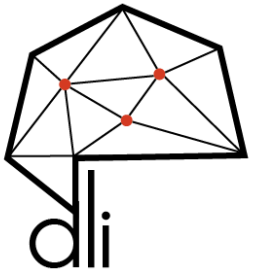
Algebrizzare lo spazio...



Le tecniche di "word embedding" provano a fare la stessa cosa con l'analisi dei testi: derivano in realtà da classiche tecniche di "information retrieval" degli anni '60 la cui idea è di codificare un documento con un punto in uno spazio di dimensione abbastanza alta.

Precisamente, si sceglie una dimensione N (alta) e si provano a "mappare" le parole di un corpus di documenti sui punti di uno spazio a N dimensioni, in modo che le relazioni semantiche fra le parole si trasformino in relazioni algebriche fra questi punti.

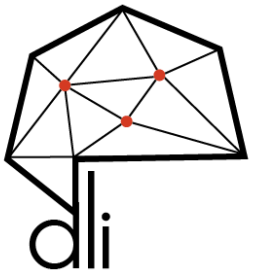
Modello cartesiano



Il principale modello matematico per qualsiasi cosa si possa immaginare è lo *spazio cartesiano*, cioè un insieme di oggetti (chiamati *punti*) che si possono sommare fra loro e moltiplicare per numeri in modo che valgano le usuali regole dell'algebra per i numeri.

Si tratta in sostanza degli spazi della geometria analitica, solo in un numero di dimensioni qualsiasi: in effetti ogni spazio cartesiano ha una ben precisa dimensione (un concetto puramente algebrico che qui non è il caso di riportare).

Lo spazio N-dimensionale

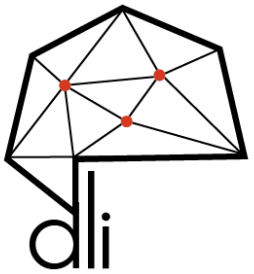


Lo spazio cartesiano di dimensione N è l'insieme $\mathbf{R}^N$ delle N-ple ordinate di numeri reali $(x_1, \ldots, x_N)$: per $N = 2, 3$ ritroviamo il concetto di piano e spazio cartesiano.

Tutta la geometria euclidea si può "mappare" sullo spazio cartesiano, e le relazioni e le costruzioni geometriche si mappano su relazioni numeriche e calcoli.

Disegno di punti nel piano cartesiano

Punti e vettori



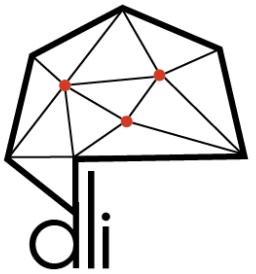
Lo spazio cartesiano $\mathbf{R}^N$ è uno spazio di punti, ciascuno dei quali è individuato univocamente dalle proprie coordinate: la differenza fra due punti Q e P si chiama un *vettore* ed è intuitivamente lo spostamento che occorre applicare a P per ottenere Q:

$$v = Q - P = \overrightarrow{PQ}$$

Anche un vettore è rappresentato da una n-pla di numeri, che si possono sommare alle coordinate di un punto P per ottenere il punto $P + v$

Cliccare qui per la demo della somma punto-vettore!

Lo spazio vettoriale



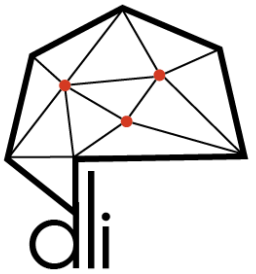
L'insieme dei vettori si chiama *spazio vettoriale*, e su di esso sono definite due operazioni "componente per componente" indotte da quelle fra numeri: se $v = (v_1, \dots, v_N), w = (w_1, \dots, w_N)$ e $\alpha \epsilon \mathbf{R}$ allora:

$$(v_1, \dots, v_N) + (w_1, \dots, w_N) = (v_1 + w_1, \dots, v_N + w_N)$$
$$\alpha(v_1, \dots, v_N) = (\alpha\, v_1, \dots, \alpha\, v_N)$$

Cliccare qui per la demo della somma!

Cliccare qui per la demo del prodotto!

Lunghezza di un vettore



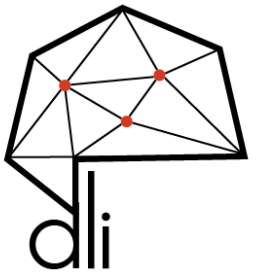
La *lunghezza* (euclidea) di un vettore v è il numero non negativo

$$|v| = \sqrt{{v_1}^2 + {v_2}^2 + \cdots + {v_N}^2}$$

Esistono altre possibili distanze (tutte topologicamente equivalenti).

Questa è direttamente ispirata al teorema di Pitagora e corrisponde, nel piano e nello spazio cartesiani, alla usuale distanza euclidea.

Distanza fra due punti



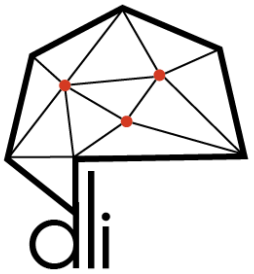
La *distanza* (euclidea) fra due punti dello spazio cartesiano P e Q è la lunghezza della loro differenza (che come sappiamo è un vettore)

$$d(P,Q) = |P - Q| = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + \cdots + (x_N - y_N)^2}$$

Se due punti hanno distanza minore di *r* allora vuol dire che uno dei due è sempre incluso in ogni "palla" di centro l'altro e raggio maggiore di *r*.

Cliccare qui per la demo!

Correlazione fra due vettori



La *correlazione* (o "similarità") fra due vettori è un numero fra -1 e 1 definito come

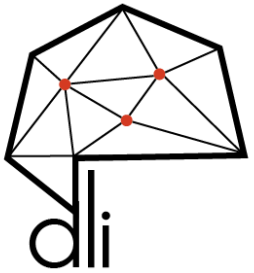
$$c(v, w) = \frac{v_1 w_1 + v_2 w_2 + \cdots + v_N w_N}{|v| \times |w|}$$

È del quoziente fra il "prodotto scalare" $v_1 w + v_2 w_2 + \cdots + v_N w_N$ per il prodotto delle loro lunghezze.

Questo indice si chiama anche (impropriamente) "distanza del coseno".

Clicca qui per la demo!

VS Model:
i documenti come vettori

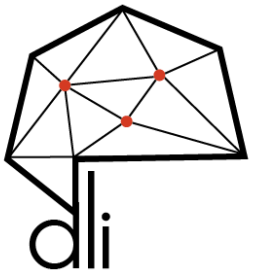


L'uso dei modelli vettoriali (*Vector Space Model*) nell'analisi dei testi risale agli anni '60 (tecniche di IR).

Negli anni '80 e '90, poggiando su queste idee, si sono utilizzati i punti degli spazi vettoriali per rappresentare documenti di testo, dove le coordinate di un documento sono dei valori associati alle parole (tipicamente basati sulla frequenza, eg. TFIDF, etc.).

Le moderne tecniche di ML si possono usare per mappare la similarità sulla semantica.

Word Embedding: le parole come vettori

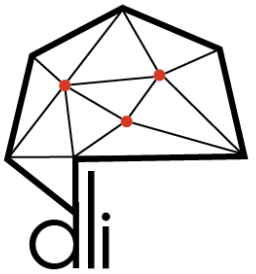


Un uso più recente degli spazi vettoriali consiste nell'usare i corpora di documenti per associare a una qualsiasi parola che occorra nel corpus un vettore in uno spazio vettoriale di dimensione N fissata.

L'idea è che una funzione {parole} → {vettori} consenta di associare a parole dal significato analogo vettori simili, o vicini.

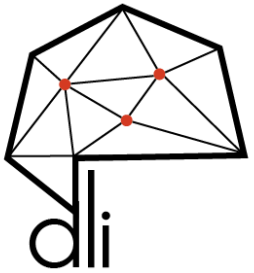
Quel che accade è che le operazioni algebriche corrispondono a operazioni semantiche: il sogno di Leibniz!!!

Agenda



- NLP: elaborazione automatica del linguaggio
- Approccio statistico alla semantica
  - Demo time!
- Modelli probabilistici: Bayes
  - Demo time!
- Semantica distribuzionale
- Modelli vettoriali: word embedding
  - Demo time!
- **Richiami sulle reti neurali**
- Un esempio: Word2Vector
  - Demo time!

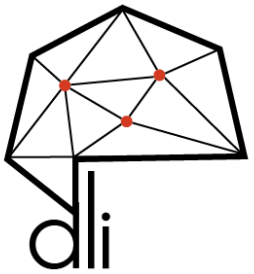
Idee vecchie, pratiche nuove



Le idee essenziali sul *machine learning* risalgono agli anni '40 (N. Wiener, W. McCulloch), '50 (R. Bellman, F. Rosenblatt) e '60 (M. Minsky), senza voler risalire ai metodi di ottimizzazione di Newton, Leibniz, Bernoulli, Eulero, Lagrange, etc. (XVIII-XIX secc.).

Ma è soltanto dalla fine degli anni ‘90 che la capacità di memoria e calcolo delle macchine consentono di sfruttarne appieno la potenza, e solo dagli anni '10 del XXI secolo che si riescono a implementare architetture "profonde". Questi algoritmi di sono tanto più efficienti quanti più dati riescono a "macinare", e oggi è disponibile una quantità astronomica di dati da elaborare.

Cosa sono le reti neurali?

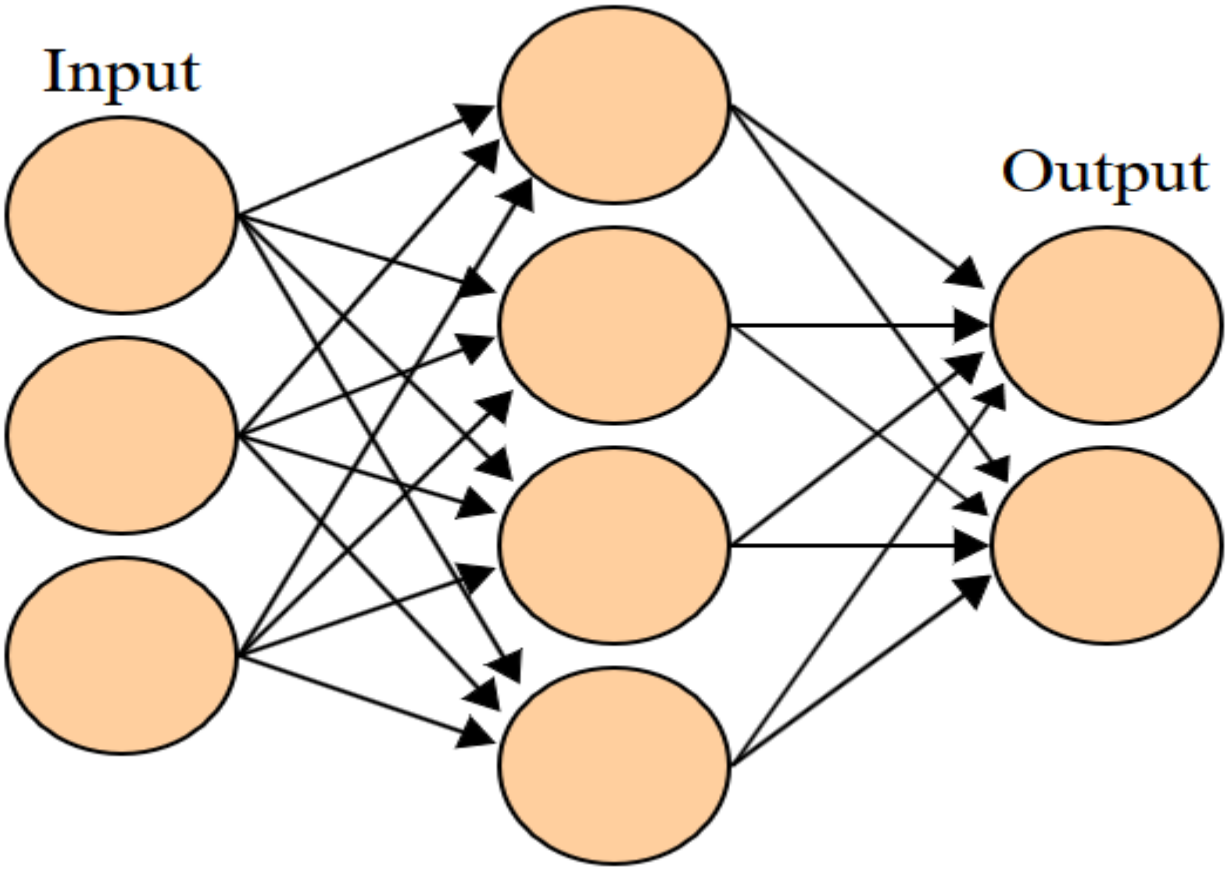


Una *rete neurale* è un particolare tipo di algoritmo iterativo di ottimizzazione che si ispira a un modello iper-semplificato del funzionamento cerebrale.

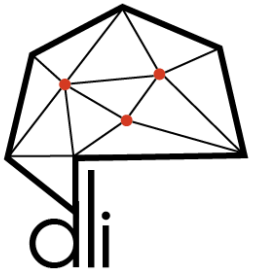
L'idea è di considerare una rete composta da una serie di neuroni disposti in "strati" (*layer*): uno strato di input prende i dati dal "mondo esterno", gli strati interni elaborano l'informazione e uno strato di output emette un risultato.

Ciascuno strato è composto da neuroni che lavorano in parallelo e che sono collegati ai neuroni dello strato precedente e dello strato seguente.

Schema di una rete neurale con uno strato interno



Come calcola un neurone artificiale

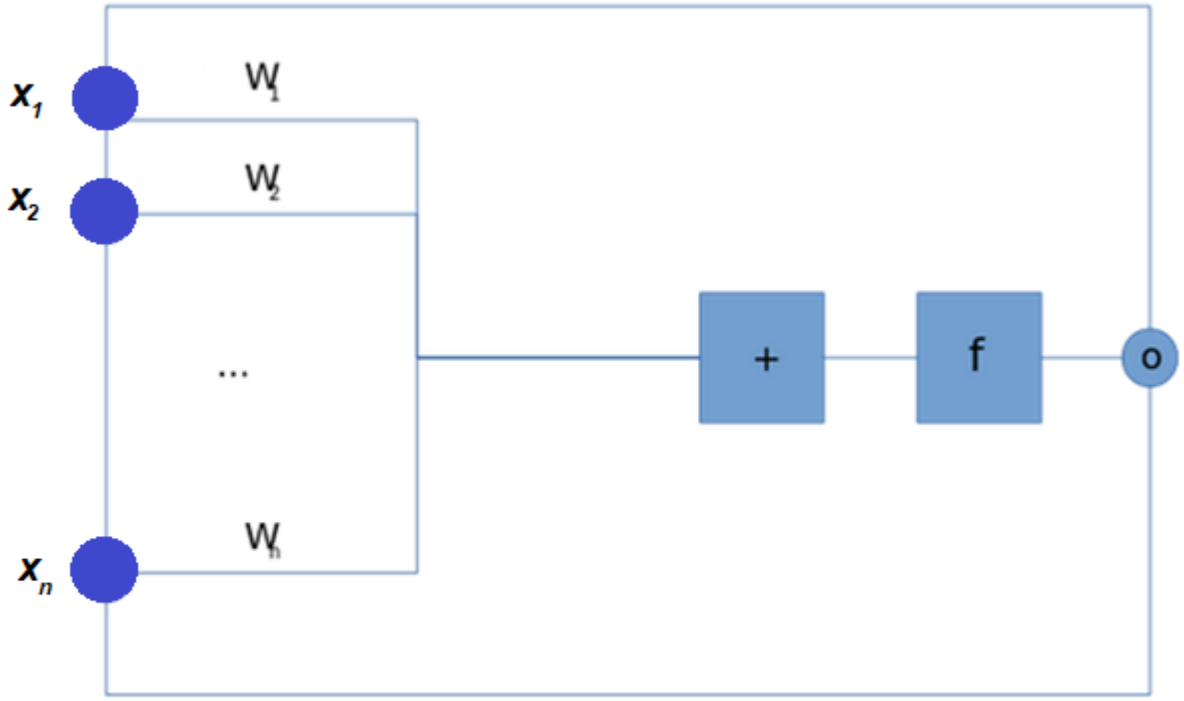


Il modello di un singolo neurone artificiale è stato ideato negli anni 1940 (!) e il suo stato interno è completamente determinato da un vettore $(w_1, w_2, \ldots, w_n)$. Il neurone accetta in input *n* numeri $x_1, x_2, \ldots, x_n$ e fornisce in output un singolo numero combinando questi numeri come segue:

$$output = f(x_1 w_1 + x_2 w_2 + \cdots + x_n w_n)$$

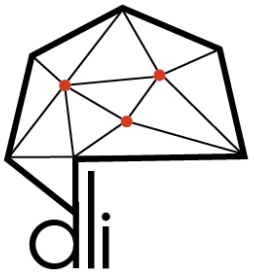
dove *f* è una funzione non lineare (si noti che pesi e input sono combinati calcolandone la correlazione come vettori).

Modello di un neurone (aka "perceptrone")



$$o = f\left(\sum_{k=1}^{n} x_k \cdot W_k\right)$$

Propagazione fra neuroni

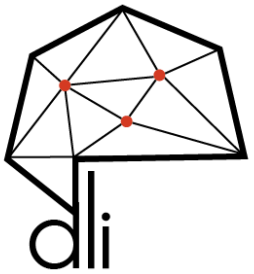


Un neurone si attiva quando le sue terminazioni di input sono stimolate dai neuroni del *layer* precedente oltre una certa soglia.

Una volta attivato, il neurone emette a sua volta un segnale in output che è il risultato di una funzione non lineare applicata ai suoi input.

In questo modo i segnali inviati allo strato di input si propagano all'interno della rete fino allo strato di output: i neuroni di quest'ultimo strato costituiscono il risultato del calcolo effettuato dalla rete.

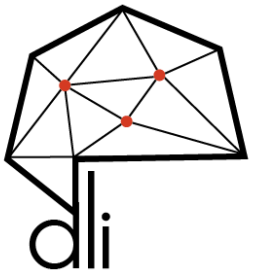
Algoritmo di calcolo di un rete



I valori dei pesi della rete di neuroni determinano come i singoli neuroni reagiscono agli stimoli di input e quale risultato producono in output: questi valori sono quindi quelli responsabili del calcolo effettivo tramite la rete.

I pesi si possono modificare e si devono modificare, in quanto la rete è generalmente inizializzata a dei valori casuali bassi nei suoi pesi, che sono poi modificati con un algoritmo chiamato "*backpropagation*" in modo da modificarsi per migliorare le proprie prestazioni nel fornire, dato un input, l'output corretto.

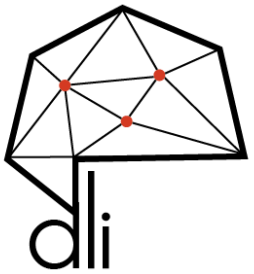
Training data



Per far svolgere a una rete un calcolo ben determinato, le si sottopongono dei *training set*, cioè insiemi di input per i quali si sa la risposta attesa, e vedere cosa risponde la rete: se la risposta della rete non è quella attesa si modificano i pesi della rete in modo da "costringerla" la prossima volta a rispondere in modo corretto per quel determinato input.

Disponendo di un training set abbastanza esteso, la rete potrà addestrarsi a riconoscere correttamente una gran varietà di input. Una volta addestrata, potrà essere usata su input per i quali la risposta non è nota.

Applicazioni delle reti

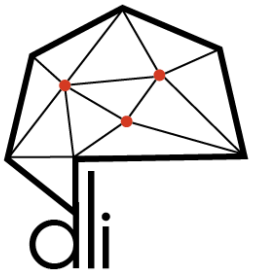


Qualsiasi problema esprimibile come approssimazione di una funzione ignota a partire dai valori che assume può essere risolto con una rete neurale.

Poiché qualsiasi informazione rappresentabile in un computer viene digitalizzata (letteralmente trasformata in numeri) è chiaro quindi come le reti neurali possano essere impiegate per riconoscere, approssimare o anche generare immagini, suoni e testi.

In particolare, come abbiamo visto, un documento di testo può essere trasformato in un vettore, quindi può essere elaborato da una rete neurale.

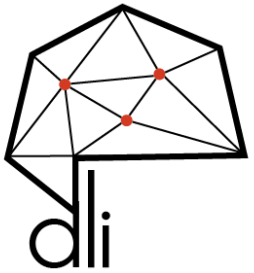
Training supervisionato e non



Se trasformiamo l'output della rete in un valore di probabilità (e ci sono varie tecniche per farlo partendo da una rete che emette semplicemente dei numeri in output), allora possiamo anche addestrarla in modo "non supervisionato" *(unsupervised).*

Più precisamente potremmo produrre un training set facendo *sampling* da una distribuzione piuttosto che etichettando manualmente i singoli documenti del training set: questo è l'approccio dell'algoritmo di word embedding che vogliamo descrivere.

Reti ricorrenti



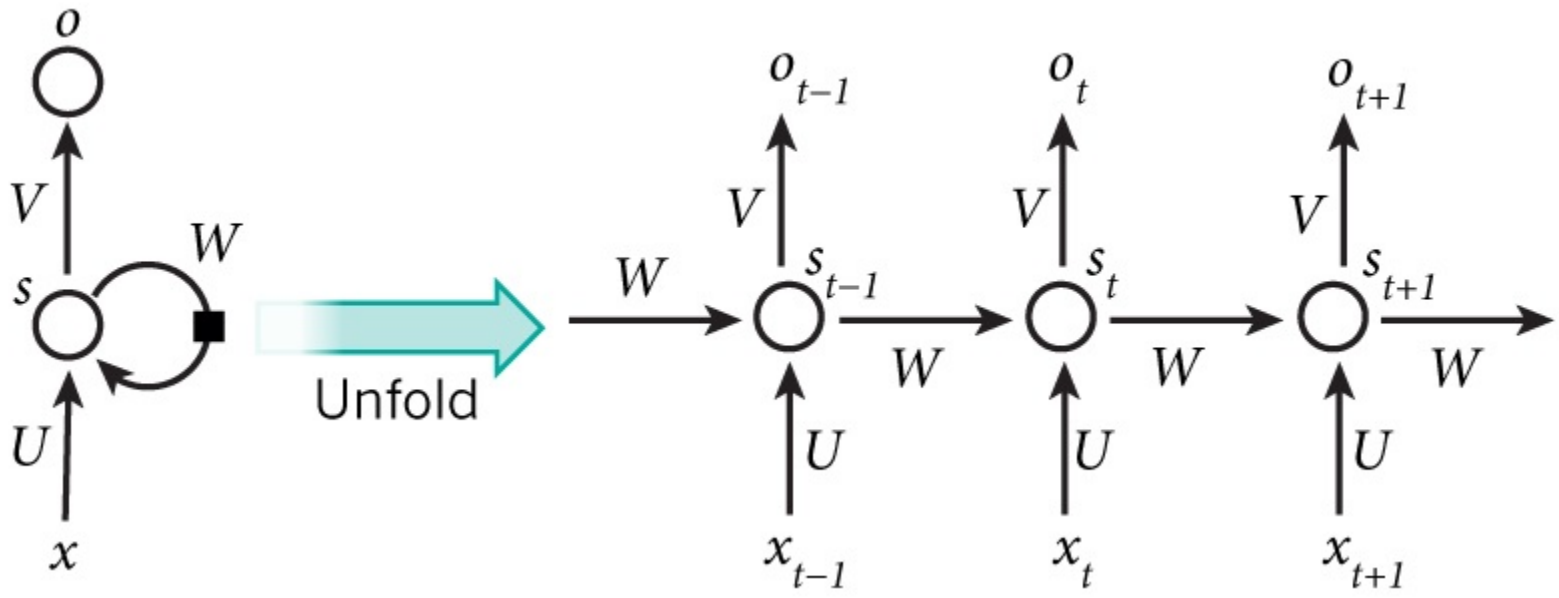
Le reti neurali "classiche" non hanno memoria, nel senso che il valore dei loro pesi al tempo t_1 dipende soltanto dal valore che avevano al tempo t_0 e, ovviamente, dall'ultimo training example.

Una *rete ricorrente* ha invece alcuni neuroni al suo interno che forniscono un ulteriore input alla rete e che sono alimentati dall'output: questi neuroni formano una sorta di "memoria storica" di quanto accaduto alla rete.

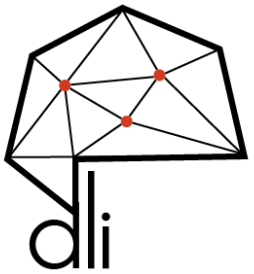
Infatti, queste reti sono usate con successo nella predizione delle serie storiche.

Reti ricorrenti

Una rete ricorrente con tre neuroni: uno di input, uno di output e uno interno che si "autoalimenta". A destra un suo "unfolding" in due istanti.



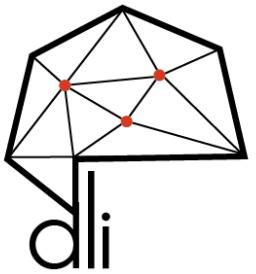
Reti ricorrenti e NLP



Le reti ricorrenti sono state impiegate con molto successo nella comprensione e anche nella generazione di testi: pensando ai nostro esempio elementare (markoviano) possiamo immaginare quanto un modello non lineare come una rete ricorrente possa migliorare le prestazioni di sistemi di generazione di testi e di classificazione e comprensione dei testi.

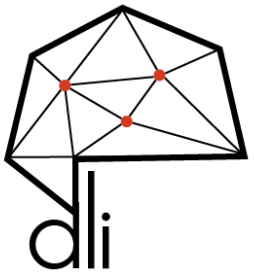
Combinando comprensione, generazione e il fatto che questi sistemi non dipendono dalla lingua, è chiaro il perché siano utilizzati per la traduzione automatica da una lingua a un'altra.

Agenda



- NLP: elaborazione automatica del linguaggio
- Approccio statistico alla semantica
  - Demo time!
- Semantica distribuzionale
- Modelli vettoriali: word embedding
  - Demo time!
- Richiami sulle reti neurali
- **Un esempio: Word2Vector**
  - Demo time!

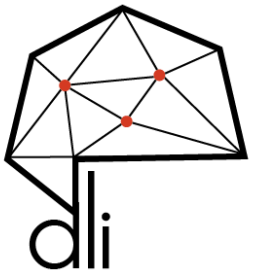
Word2Vector



Veniamo alla rete di nostro interesse: Word2Vector. Si tratta di una rete neurale ricorrente con un layer interno, che viene usata in una modalità non supervisionata.

In pratica, per ogni parola del corpus si produce un vettore binario *w*, di dimensione pari alla cardinalità *N* del corpus, che rappresenta la parola e lo si usa come input per la rete, che in output fornisce una distribuzione di probabilità *y* sullo "spazio" delle parole, dunque ancora un vettore di dimensione *N*.

Word2Vector

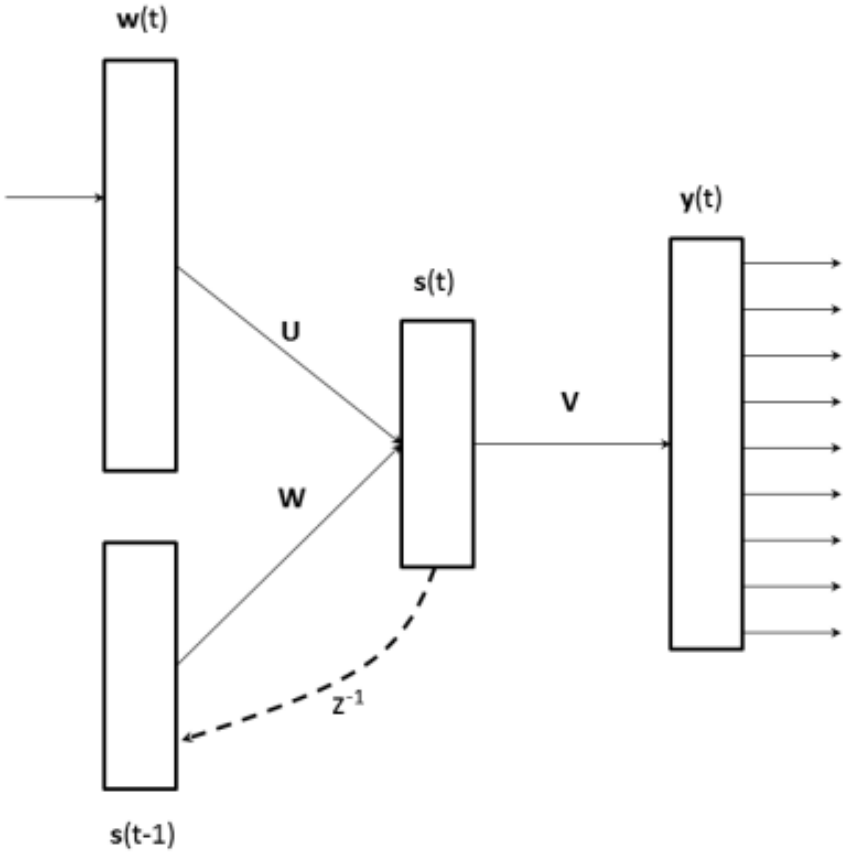


La rete impara il legame che esiste fra la distribuzione di probabilità di una parola e il contesto distribuzionale nel quale appare.

Si noti che l'approccio markoviano di dedurre una parola data la precedente è un caso lineare e particolare di questi due.

Il layer interno della rete neurale accumula dei valori prodotti via via che le parole del testo sono presentate alla rete, e conserva quindi memoria dei contesti al cui interno si manifestano le parole esaminate dalla rete: questo è l'elemento cruciale dell'algoritmo.

Architettura della rete

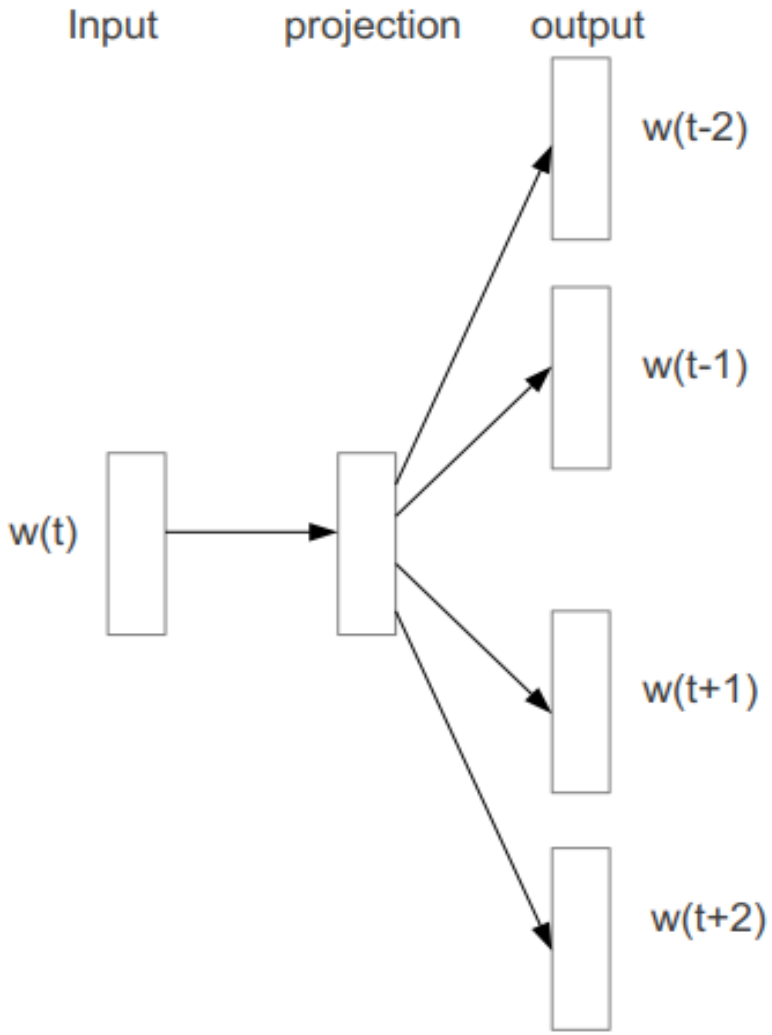


$s(t) = Uw(t) + Ws(t-1)$

$y(t) = Vs(t)$

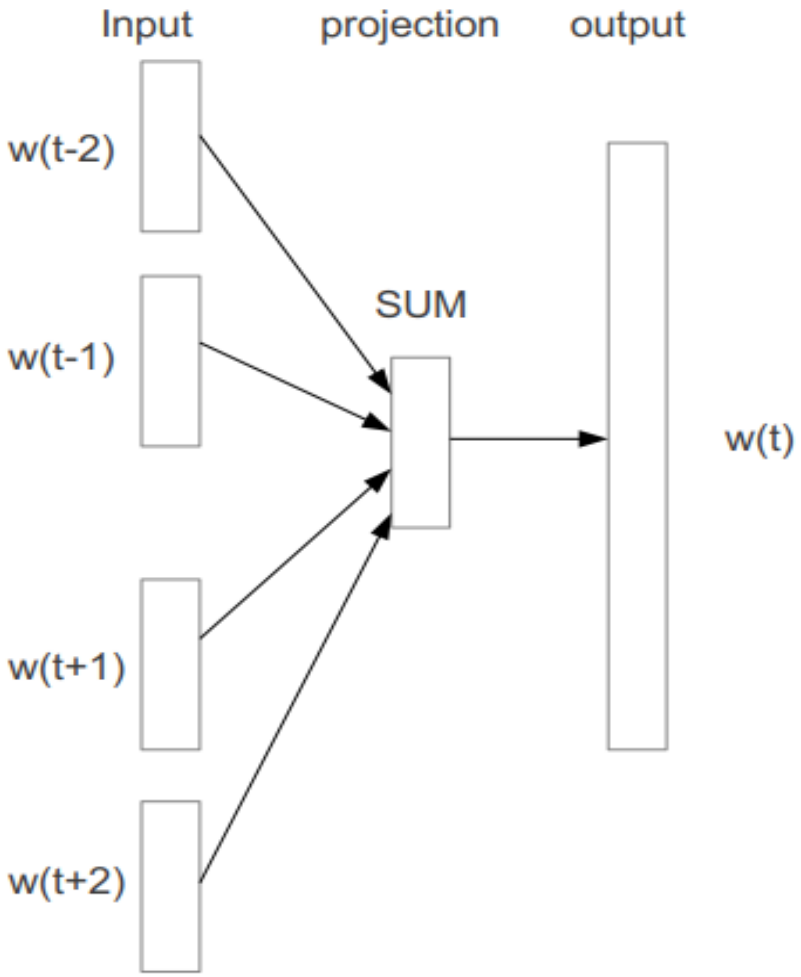
Sia y che s sono filtrati tramite delle funzioni: con la rappresentazione "soft-max" il primo, con una sigmoide f(z) il secondo.

Addestramento Skip Gram



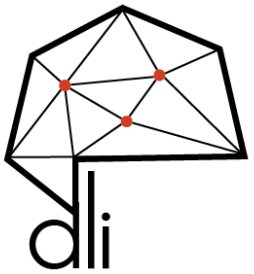
La rete si addestra a produrre la distribuzione di probabilità *y* in modo che dalla parola di input si possa predire il contesto

Addestramento CBOW



La rete si addestra a produrre la distribuzione di probabilità *y* in modo che dal contesto di input si possa predire la parola (cioè si completi il contesto con una parola plausibile).

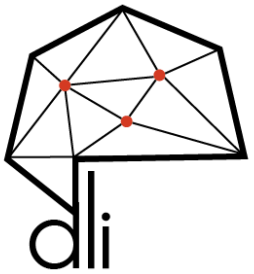
I vettori associati alle parole



I vettori associati alle parole, cioè il reale output che interessa per il modello semantico distribuito, sono estratti come colonne dalla matrice *U* che fornisce il collegamento fra il *layer* di input e il *layer* interno della rete neurale.

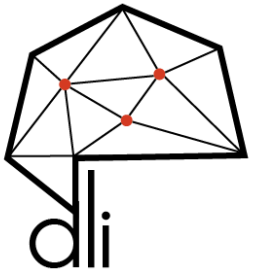
Questo algoritmo può essere implementato in modo efficiente e, per corpora abbastanza vasti (almeno miliardi di parole), produce risultati molto accurati migliorando di molto le prestazioni dei precedenti algoritmi di analisi semantica distribuita.

Agenda



- NLP: elaborazione automatica del linguaggio
- Approccio statistico alla semantica
  - Demo time!
- Modelli probabilistici: Bayes
  - Demo time!
- Semantica distribuzionale
- Modelli vettoriali: word embedding
  - Demo time!
- Richiami sulle reti neurali
- Un esempio: Word2Vector
  - **Demo time!**

Esempio d'uso

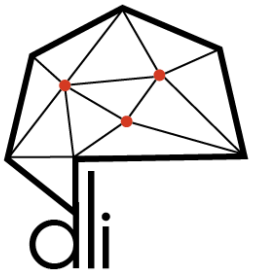


Proviamo a visualizzare i punti che word2vec associa alle parole: per farlo utilizziamo un insieme di vettori (con 200 componenti) che rappresentano circa 70K parola, prodotti dalla rete neurale su un corpus di documenti presi da pagine di Wikipedia (en).

Nella seguente demo inserire una lista di parole che il programma assocerà ai vettori corrispondenti, proiettandoli sul piano cartesiano (con l'algoritmo t-SNE che tenta di preservare le distanze):

Cliccare qui per la demo

Punti = parole
Vettori = concetti



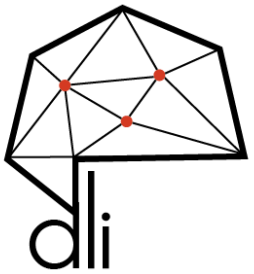
Un aspetto che ha del miracoloso di questo algoritmo è che alcune relazioni semantiche si ottengono con operazioni fra i vettori delle parole corrispondenti.

Per esempio: se scriviamo *v(p)* per il vettore associato alla parola *p*, e *p(v)* per la parola corrispondente al *vettore più vicino* al vettore *v*, troviamo che

p(v("king") - v("man") + v("woman")) = "queen"

Cioè il vettore "woman" - "man" corrisponde al concetto di "femminilità"...

Algebra della semantica



Altri esempi portati dai creatori di word2vec su un dataset 1000 volte più grande del nostro:

p(v("bigger") - v("big") + v("cold")) = "colder"

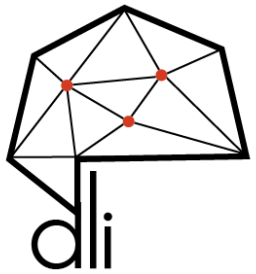
p(v("Paris") - v("France") + v("Italy")) = "Rome"

p(v("sushi") - v("Japan") + v("Germany")) = "bratwurst"

p(v("Windows") - v("Microsoft") + v("Google")) = "Android"

Provare per credere!!!

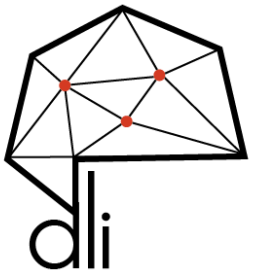
Parole simili



Un altro semplice esperimento che è possibile provare è, data una parola, trovare quelle che le sono più vicine, per esempio le 10 più vicine:

Trova le parole simili a una parola data

Grazie per l'attenzione!!!



Credits

Wikipedia per le immagini.

Tomas Mikolow per l'algoritmo, il codice e i paper di riferimento (da alcune sue slide abbiamo preso delle figure).

Google Code per il codice di Word2Vector.

Matt Mahoney per il dataset utilizzato per word2vec.

http://www.caressa.it

https://www.linkedin.com/in/paolocaressa

@www_caressa_it