Linguistica Computazionale

Corpus

Per definizione: un'opera di selezione per tipo

Un corpus, può essere rappresentativo per:
- Contesto: ossia contenente token specifici per un determinato ambito
- Lingua: ossia contenente token di diversi campi affinché copra il più possibile una lingua

Generalmente un corpus non copre interamente la lingua

Parametri di un corpus

Generalità
generali o specialistici

Modalità:
scritto, audio o misto

Cronologia:
diacronico o sincronico

Lingua:
- monolingua
- paralleli (più lingue)
- comparabili

Integrità dei testi

Livello di codifica digitale dei testi

Un corpus può essere anche classificato per token contenuti.
"*The larger, the better*"

Anche l'intero web può essere utilizzato come corpus, ma non totalmente affidabile per via di dati possibilmente errati (*rumore*)

Corpus principali conosciuti

Brown Corpus, 1961
US English
Circa un milione di token

British National Corpus (BNC), misto
British English
Circa 100 milioni di token, 90% scritto e 10% orale

La Repubblica
Italiano
Circa 326 milioni di token
Tema generale, specialistico nella tipologia testuale: scritto, monolingua e annotato

Linguistica Computazionale

Codifica del testo

Per poter analizzare digitalmente un testo è necessario trasformarlo in un formato detto Machine Readable Form (MRF)

Esistono principalmente due livelli di codifica del testo

- Codifica livello 0: Definito anche come *plain text* (testo "piano"). Utilizza la semplice rappresentazione binaria dei caratteri in *ASCII* (pron. /'æski/), *ISO, UTF, ...*
- Codifica livello 1: Vengono aggiunte, al livello 0, informazioni utili per l'interpretazione del testo (*metadati*) tramite linguaggi di mark-up (es: XML)

La preparazione del testo da analizzare segue dei passaggi ben specifici:

- *text cleaning*: pulizia da errori vari (grammaticali, sintassi, etc)
- *verifica dei caratteri*: si tende a utilizzare una rappresentazione univoca dei caratteri, generalmente *UTF-8*
- *normalizzazione*: modificare tutti i caratteri maiuscoli in minuscolo, ma non sempre (possono creare ambiguità)
- *word-segmentation*: tokenizzare il testo, laddove possibile (lingue non segmentate come le l. asiatiche possono comportare errori di suddivisione)

Analisi statistica del testo

Una volta eseguita la corretta tokenizzazione del testo, possono essere svolti dei compiti preliminari per l'analisi statistica del testo

Calcolo delle *Frequenze Assolute* dei token (quante volte un certo token appare nel corpus)

(tokens) $|C| = |f_{(V0)}|+|f_{(V1)}|+...+|f_{(Vn)}|$
La somma di tutte le frequenze assolute è pari alla lunghezza intera del corpus C

(types) $|V| = v_0+v_1+...+v_n$
La somma di tutti i token, presi una sola volta, costruisce il vocabolario V del corpus C. Vengono essi definiti *types*.

Tramite questi due valori, è possibile calcolare l'indice di ricchezza lessicale, chiamato *Token-Type Ratio* (TTR), e definito come **TTR = |V|/|C|**

Il TTR è un valore sempre compreso tra 0 e 1.

- TTR == 0 → |C|==0 (è un "corpus" vuoto)
- 0<TTR<1 → |C|!=|V|
- TTR == 1 → |C| == |V| (il corpus è composto solo da *hapax legomenon*, token con frequenza pari a 1)

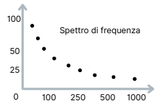
Frequenza Relativa dei token
La frequenza relativa serve a indicare quanto, un determinato token, copre un corpus. Si calcola come $F(v_i) = f(v_i)/|C|$
Se moltiplichiamo $F(v_i)$ • 100 si ottiene il valore in percentuale (es: v_i copre C al x%)

La *Frequenza Relativa Cumulata* indica la frequenza totale su parte di C

Classi di Frequenza
Si raggruppano i *types* con la stessa frequenza e si calcola come $v_i \bullet i$
i indica il rango (es: types con freq. 5)
v_i indica le classi
Sommando tutte le classi di frequenza, si ottiene |C|

$$|C| == |v_i| \bullet i$$

Rappresentabile graficamente con lo spettro di frequenza (grafico →)



Linguistica Computazionale

Analisi statistica del testo

Anche se i vocabolari di due corpus possono coincidere al 10%, essi potrebbero contenere una percentuale maggiore di tokens uguali.

Anche se i vocabolari di due corpus possono coincidere al 10%, essi potrebbero contenere una percentuale maggiore di tokens uguali.

Ciò è dovuto dalla presenza di parole dette *piene* e *grammaticali*

Vengono considerate *piene* le parole contenenti significato: *casa, animale, …*
Queste si presentano come: molti types, bassa frequenza (*LNRE*, Large Number of Rare Events)

Sono considerate *grammaticali* le parole prive di significato, come gli articoli, congiunzioni, preposizioni, ecc.
Queste si presentano come: pochi types, alta frequenza

La frequenza media delle parole all'interno di un corpus C deriva dalla definizione di media aritmetica:

$$\bar{f} = \Sigma f(v_i) \,/\, |V|$$

Considerando che $\Sigma f(v_i) == |C| \rightarrow \bar{f} = |C|/|V|$ (TTR inversa)

L'andamento tendenziale della frequenza media $\bar{f}$, al crescere dei campioni k, è una funzione <u>non decrescente</u> $\bar{f}_k \leq \bar{f}_{k+1}$

Esso può variare al variare del vocabolario, presentando un segno di discontinuità

Il rango r_v di una parola è la posizione che essa occupa in base alla sua frequenza, dal maggiore al minore.

La legge di Zipf stabilisce che la frequenza di una parola è inversamente proporzionale al suo rango: $f_z = C \,/\, z^a$

- f_z è la frequenza della parola di rango z
- C è una costante e corrisponde a f_1
- z indica il rango
- a è una variabile che viene approssimata a 1

La legge di Zipf prevede un decremento progressivo della frequenza di una parola, proporzionale all'aumento del suo rango

In generale, al crescere del rango, diminuisce lo scarto (C/n)-(C/n+1) tra le frequenze di due parole di ranghi adiacenti

Conseguenze

Vocabolario aperto: tendenzialmente un testo non copre l'intero vocabolario di una lingua

Data sparseness: la maggior parte delle parole hanno frequenze basse

Promemoria proprietà logaritmiche: (tendenzialmente si intende $\log_{10}$)

- $a \text{ ^ } \log_a b = b$
- $\log(b \bullet c) = \log b + \log c$
- $\log(b/c) = \log b - \log c$
- $\log(b^c) = c \log b$

L'applicazione logaritmica alla legge di Zipf rende un rapporto costante in differenze costanti: $\log f_z = \log C \,/\, z^a \rightarrow \log C - a \log z$

Ciò appiattisce, graficamente, la linea dello spettro di frequenze

Linguistica Computazionale

Analisi probabilistica del testo

Un modello probabilistico di un evento determina la probabilità che tale evento si verifichi oppure no (*evento aleatorio*).

Un corpus di testi si usa per:
- calcolare le statistiche sulle distribuzioni delle strutture linguistiche
- costruire un modello che assegna una probabilità a degli eventi

Natural Language Processing (*NLP*)

- Maggiore robustezza in mancanza di informazioni
- Capacità di generalizzazione
- Approccio induttivo ed empireo, stimando a partire dai *training corpus*

Probabilità

La stima frequentista della probabilità dei parametri di un modello probabilistico viene chiamata *Maximum Likelyhood Estimation* (MLE).
Tende a massimizzare la probabilità di *v* nel linguaggio.

Il fenomeno del *data sparseness* tende a non rendere affidabile la stima probabilistica in caso di frequenze estremamente basse.

Per risolvere il problema si usa la tecnica dello *smoothing* (smussamento), o anche detto *Add-One Smoothing* o Legge di Laplace.

La legge di Laplace permette di stimare eventi non presenti nel corpus, i quali potrebbero azzerare le probabilità complessive.
$\mathbf{P(e) = f_e / |C|}$ – Laplace → $\mathbf{P_s(e) = (f_e + 1) / (|C| + |V|)}$

La probabilità è un numero che indica il grado di incertezza al verificarsi di un evento A (*aleatorio* = che può avvenire oppure no) e si calcola come $\mathbf{P_A = f_A / \Omega}$ dove f_A indica la frequenza relativa di un dato evento (un token) e Ω indica lo spazio campione (il corpus |C|)

In Linguistica Computazionale, la formula della probabilità viene "tradotta" come $\mathbf{P_V = f_V / |C|}$

Assiomi della probabilità

- $\mathbf{P_A \geq 0}$ comunque preso A → la probabilità non può mai essere un valore negativo
- $\mathbf{P_{AuB} = P_A + P_B}$ se l'intersezione di A e B è l'insieme vuoto → regola della somma
- **P(Ω) = 1** → certezza
  NB: la probabilità di A non è un evento certo. Esso è un evento certo se e solo se A == Ω, ossia un corpus formato da un solo token

Linguistica Computazionale

Analisi probabilistica del testo

Probabilità congiunta

$P_{(A,B)}$ è la probabilità che due eventi A e B si verifichino assieme

Esso si calcola su eventi indipendenti. Due eventi si dicono indipendenti se il verificarsi di A non implica il verificarsi di B. In questo caso vale che $\mathbf{P_{A,B} = P_A * P_B}$

Probabilità condizionata

$P_{(B|A)}$ è la probabilità che si verifichi l'evento B dopo che è avvenuto l'evento A.

In questo caso si calcola su eventi dipendenti: sapendo che è avvenuto A, ciò altera la probabilità di B, dunque $\mathbf{P_B \neq P_{(B|A)}}$

Per calcolare la probabilità condizionata si utilizzano le frequenze dei bigrammi, ossia le frequenze che una sequenza di token A-B sia avvenuta.

$$\mathbf{P_{(B|A)} = f_{<A,B>} / f_{<A,Vi>}}$$

$f_{<A,Vi>}$ indica la frequenza di tutti i bigrammi in cui compare A, dunque si semplifica in $f_{A,}$ trasformando la formula in $\mathbf{P_{(B|A)} = f_{<A,B>} / f_A}$

In generale vale che:

- se $P_A \neq 0$, allora $P_{(B|A)} = P_{(A,B)} / P_A$
- se $P_A \neq 0$, allora $P_{(A,B)} = P_A * P_{(B|A)}$
- se A e B sono indipendenti e $P_{(B|A)} = P_{A,}$ allora $P_{(A,B)} = P_A * P_{(B|A)} \rightarrow P_B * P_A$

Language Models

I language models sono dei modelli che assegnano una probabilità ad una sequenza di parole $P(v_1, v_2, \ldots, v_n)$, chiamati *n-gram models*.

Uno dei campi in cui si applicano gli n-gram models è quello della *word-prediction*: con quale probabilità, data una sequenza $v_1, \ldots, v_n$, sarà presente un elemento v_{n+1} (probabilità condizionata $P(v_{n+1}|v_1,\ldots,v_n)$, elemento v_{n+1} dato un n-gramma)

Dato il rapporto tra probabilità condizionata e congiunta, equivale a trovare la parola che massimizza il valore di $P(v_1, \ldots, v_n, v_{n+1})$.

Per stimare la probabilità di n-grammi si usava la *chain rule*:

$\mathbf{P(v_1,\ldots,v_n) = P(v_1) * P(v_2|v_1) * \ldots * P(v_{n+1}|v_1,\ldots,v_n)}$

Ci permette di calcolare la probabilità di sequenze di caratteri, parole, frasi.

contro

Non sono riconosciute le frasi anagrammaticali

Su sequenze brevi può essere comodo, ma su sequenze lunghe potrebbero trovarsi casi che comprometterebbero i calcoli, ad esempio un elemento con bassa frequenza, tendente a 0.

Le probabilità di sequenze di n-grammi si possono misurare tramite le frequenze degli n-grammi stessi:

$$\mathbf{P(v_n|v_1,\ldots,v_{n-1}) \approx F(v_1,\ldots,v_n) / F(v_1,\ldots,v_{n-1})}$$

Linguistica Computazionale

Analisi probabilistica del testo

Modello di Markov

Questo modello probabilistico semplifica i calcoli su sequenze di caratteri, parole o frasi, limitandoli a semplici n-grammi che non invalideranno il valore finale. In breve, se vogliamo conoscere la probabiltà di E_i, basterà conoscere la "storia" che precede.

Si suddivide in ordini, in base alla dimensione degli n-grammi considerati. Generalmente si considerano i primi tre.

Modello Markov 0
Il livello 0 presuppone l'indipendenza degli elementi, si calcola come una semplice probabilità congiunta:

$$P(v_1,\dots,v_n) = P(v_1)*\dots*P(v_n)$$

Modello Markov 1
Il livello 1 presuppone la dipendenza degli elementi, si calcola come una probabilità condizionata su bi-grammi:

$$P(v_1,\dots,v_n) = P(v_1)*P(v_2|v_1)*\dots*P(v_n|v_{n-1})$$

Modello Markov 2
Il livello 2 presuppone la dipendenza degli elementi, si calcola come una probabilità condizionata su tri-grammi:

$$P(v_1,\dots,v_n) = P(v_1)*P(v_2|v_1)*P(v_3|v_1,v_2)*\dots * *\dots*P(v_n|v_{n-1},v_{n-2})$$

Per tutti i modelli superiori al livello 2, la regola si espande in base all'n-gramma considerato.
Il livello massimo si espande fino a diventare una *chain rule, diventando così "useless" ai fini di Markov*

$$P(v_1,\dots,v_n) = P(v_1) * P(v_2|v_1) * \dots * P(v_{n+1}|v_1,\dots,v_n)$$

Le catene di Markov permettono di creare dei modelli probabilistici di sequenze linguistiche per le quali esistono particolari tipi di dipendenza.

Utilizzando un algoritmo addestrato con catene di Markov su un training corpus, è possibile migliorare la generazione di un testo aumentando le dipendenze ad ogni ordine.

Utili per molte competenze del *NLP* come l'*Automatic Speech Recognition*, il *Machine Transition* e soprattutto la *word-prediction*, simile al *MLE*

In casi di elementi con basse frequenze, è comunque possibile applicare Laplace (*Add-One Smoothing*)

Linguistica Computazionale

Combinazioni

Alcune parole sono legate ada ltre tramite forti lessicazioni morfologiche e semantiche. Esse possono essere sostituite con altre parole per ottenere altre frasi grammaticali.

Esistono altri tipi di combinazioni che si basano su legami non riconducibili a classi linguistiche generali. Sono difficili da sostituire perché produrrebbero strani risultati.

Combinazioni di due o più parole caratterizzate da un elevato grado di associazione, determinata dalla tendenza di co-occorrere:

- argomenti o modificatori “tipici”
- argomenti o modificatori “idiosincratici”
- costruzioni idiomatiche
- nomi propri composti
- ...

Collocazioni

Distinguibiili in due categorie

Empirico, o senso ampio: combinazioni ricorrenti e predicibili di parole, osservate nell’uso linguistico (*corpora*)

Teorico, o senso stretto: espressioni polirematiche fortemente lessicalizzate, idiomatiche e idiosincratiche (*multiword expressions*)

Sono parole con un alto grado di associazione reciproca. Le misure di associazione lessicale:

Quantificano la forza del legame tra due o più parole sul piano sintagmatico

La nozione intuitiva di associazione lessicale viene trasformata in un indice quantitativo e misurabile

Elevata convenzionalità
Sono tendenzialmente espressioni di usi convenzionali, tipici di varietà linguistica

Ridotta composizionalità semnatica
Non immediatamente ricavabile dalla composizione delle parole che lo formano, ad esempio:
{topolino grigio} == {topolino} + {grigio}
{gatta morta} != {gatta} + {morta}

Forte rigidità strutturale
Sono spesso resistenti a modificazioni aggettivali o avverbiali, oppure occorrono solo in particolari forme flesse e contesti sintattici.

Linguistica Computazionale

Collocazioni

1. Analisi linguistica del corpus: il testo deve essere tokenizzato e possibilmente annotato con PoS (*Part of Speech*) tagging, lemmatizzazione, ecc.
2. Selezione dei bigrammi: il tipo dei bigrammi che vengono selezionati dipende dal livello di annotazione.
3. Costruzione della tabella di contingenza: dal totale dei bigrammi costruisco la tabella.
4. Applicazione di una misura di associazione
5. Ordinamento delle coppie in base alla forza di associazione.

Preso il bigramma <x,y> avremo

	x	!x	
y	O_{11}	O_{12}	R_1
!y	O_{21}	O_{22}	R_2
	C_1	C_2	

- O_{11}: bigramma dove occorrono x e y
- O_{12}: occorre y, ma non x
- O_{21}: occorre x, ma non y
- O_{22}: non occorrono né x né y

$N = R_1 + R_2 == C_1 + C_2$

Partendo dalla tabella di contingenza:

	x	!x
y	E_{11}	E_{12}
!y	E_{21}	E_{22}

- E_{11}: $(R_1 \bullet C_1)/N$
- E_{12}: $(R_1 \bullet C_2)/N$
- E_{21}: $(R_2 \bullet C_1)/N$
- E_{22}: $(R_2 \bullet C_2)/N$

Due parole si dicono fortemente associate quando si presentano insieme più spesso rispetto alle singole frequenze.

È necessario confrontare la frequenza osservata O di <x,y> con la sua frequenza attesa E, ossia la frequenza che ci aspettiamo qualora gli elementi fossero statisticamente indipendenti.

Con entrambe le tabelle, è possibile calcolare la *Mutual Information* (MI):

$$\mathbf{MI<x,y> = \log_2 O<x,y>/E<x,y>}$$

Se la MI ≤ 0, c'è assenza di associazione tra le parole, altrimenti potrebbe essere una forte associazione.

La *Local Mutual Information* (LMI) privilegia i bigrammi più frequenti ed è il termine fondamentale nel calcolo del *Log Likelyhood Ratio*

$$\mathbf{LMI<x,y> = f<x,y> \bullet MI<x,y>}$$

Un'altra versione con cui si calcola la MI in termini di probabilità:

$$MI<x,y> = \log_2 P_{X,Y}/(P_X P_Y) == \log_2 (f<x,y>/N)/((f_X/N)(f_Y/N)) \rightarrow$$
$$\rightarrow (f<x,y> \bullet N)/(f_X f_Y)$$

Estremamente sensibile agli eventi rari. In qualsiasi corpus i bigrammi formati da hapax avranno valori massimi.

Se x e y ricorrono sempre insieme:

$$\mathbf{MI<x,y> = f * N/f^2 = N/f}$$

Linguistica Computazionale

Legge di Bayes

Viene usata per decidere qual è l'ipotesi I, o classe, più probabile all'interno di IP che spiega un certo tipo di osservazioni O

$$\textbf{argmax}_{I \in IP} P_{(I|O)} = \textbf{argmax}_{I \in IP} (P_I * P_{(O|I)})/P_O$$

Poiché stiamo cercando l'ipotesi più probabile (argmax), data la stessa osservazione O, possiamo ignorare P_O:

$$\textbf{argmax}_{I \in IP} P_{(I|O)} = \textbf{argmax}_{I \in IP} P_I * P_{(O|I)}$$

Ci permette di tradurre $P_{(I|O)}$ in un prodotto di probabilità più facile da stimare.

Chiamato *teorema della probabilità delle cause*, viene impiegato per calcolare la probabilità di una causa che ha provocato l'evento verificato.

Molti fenomeni linguistici possono essere modellati come processo di inferenza bayesiana, come:

- Riconoscimento del parlato come l'*Automatic Speech Ricognition* (ASR)
- Problema di decidere a quale classe appartiene una certa osservazione linguistica

Deriva della definizione di probabilità condizionata

$$\textbf{P}_{(A|B)} = \textbf{P}_{<A,B>}/\textbf{P}_B \rightarrow \textbf{P}_{<A,B>} = \textbf{P}_B * \textbf{P}_{(B|A)}$$

$P_{(A|B)} = P_{<A,B>}/P_B = P_{(B|A)}P_A / P_B$ dove:

- P_A → probabilità a priori
- $P_{(B|A)}$ → likelyhood
- $P_{(A|B)}$ → probabilità a posteriori

Noisy Channel Mode

Un sistema dove si introduce un input in un canale "rumoro", alternandone l'output. Il canale rumoroso contiene gli elementi dell'inferenza bayesiana, che produce in output un'ipotesi con una certa probabilità.

Linguistica Computazionale

Entropia

Mentre le probabilità misurano fenomeni e sistemi il cui esito è incerto, l'entropia è una misura dell'incertezza di un fenomeno e misura la difficoltà nel predire l'esito.

- L'incertezza dipende da:
  - numero di esiti alternativi possibili.
  - distribuzione delle probabilità per ciascun esito; in caso di esiti con probabilità uniforme è più difficile prevedere quello esatto.
- L'informazione è la diminuzione dell'incertezza: se un evento aleatorio si verifica, avremo informazioni sia sugli eventi positivi che negativi.
  - Più un evento è raro, più è informativo, maggiore è l'informazione che otteniamo col verificarsi.
- L'entropia è la misura della quantità dell'informazione portata dagli eventi prodotti da eventi aleatori, formalizzato come una variabile casuale.
  - L'unità di misura è il bit.
  - L'entropia aumenta con l'aumentare degli stati possibili di un sistema. A parità di stati possibili, l'entropia diminuisce se aumenta la "struttura" e l'organizzazione del sistema, e aumenta la predicibilità delle dinamiche del sistema.
  - Una *Random Variable* rappresenta un processo aleatorio, il quale è descritto dall'insieme di stati che esso può assumere e da una distribuzione di probabilità associata.
- Il codice Shannon-Fano dice che: gli stati più probabili e frequenti sono descritti usando messaggi più corti, gli stati meno probabili usano messaggi più lunghi.

L'*entropia puntuale*, o informazione di una parola, corrisponde al numero di bit necessari per trasmettere che è stata estratta quella parola.
Si calcola come:

$$h_V = - \log_2 P_V$$

Se un sistema W ha m stati equiprobabili, il numero di bit necessari per codificare ogni stato è $\log_2 m$.
Se gli stati sono equiprobabili vale che $P_V = 1/m$, di conseguenza $m = 1/P_V$. Per le proprietà logaritmiche avremo che:

$$h_V = \log_2 m \rightarrow h_V = \log_2 1/P_V = \log_2 1 - \log_2 P_V \rightarrow h_V = - \log_2 P_V$$

- Nel caso di stati *non* equiprobabili, l'entropia di un sistema W è definita come:

$$H_W = -\sum P_{Vi} \log_2 P_{Vi}$$

- Date le distribuzioni in due stati:

$$H_W = -\sum P\langle w,t_i\rangle \log_2 \langle w, t_i\rangle$$

La distribuzione minore è quella più caratterizzante.

L'entropia *per sequenze di parole* si misura come:

$$H(w_i) = -\sum P(w_i) \log_2 P(w_i)$$

L'entropia *per parola*, o *entropy rate*, si misura come:

$$H_{rate}(w_i) = 1/n\ H(w_i)$$

Linguistica Computazionale

Entropia

Shannon-McMillan-Breiman: se un linguaggio L è generato da un processo *stocastico ergodico* e *stazionario*, allora vale che:

$$\mathbf{H_{rate}\ L = \lim_{n\to\infty} -1/n * \log_2 P(w_1,\ldots,w_n)}$$

Un linguaggio è *stazionario* se la probabilità che assegna a sequenze di parole sono invarianti rispetto al tempo.

Un linguaggio è *ergodico* se, aumentando la lunghezza della sequenza di parole generate, possiamo ottenere un campione perfettamente rappresentativo del processo.

In realtà, però, il linguaggio non è né stazionario né ergodico.

Cross Entropy: $\mathbf{H(w_1, m) = -\sum P(w_1) \log_2 P(w_1)}$

- w_1 è una variabile casuale con distribuzione reale P
- m è un modello stocastico di w_1 che cerca di appossimare la sua distribuzione reale.

La cross entropy ci consente di misurare quanto bene un modello probabilistico approssima un certo processo stocastico $\mathbf{H(w) \leq H(w,m)}$
È il costo in bits per usare m come modello descrittivo di un processo con distribuzione P.

Per Shannon-McMillan-Breiman è possibile approssimare la cross entropy prendendo un campione sufficientemente grande di testi del linguaggio come unica sequenza di parole

$$\mathbf{H(p,m) = \lim_{n\to\infty} -1/n * \log_2 m(w_1,\ldots,w_n)}$$

Il modello più accurato sarà quello con la cross entropy minore.

Natural Language Processing

Il *Natural Language Processing* è un sistema in grado di accedere al contenuto di informazioni attraverso l'elaborazione del linguaggio.

Vengono effettuti delle task tipiche della preparazione per l'analisi statistica.

Esistono diversi problemi che possono presentarsi nell'addestramento degli algoritmi per via delle ambiguità fonologiche, morfologiche, sintattiche, semantiche, etc.

Il *Machine Learning* comprende gli algoritmi che permettono alla macchina di imparare a svolgere un compito X, partendo da degli esempi su come svolgere quel determinato compito.

Linguistica Computazionale

Machine Learning

Usa modelli statistici dei dati nel corpus al fine di costruire un modello per svolgere il lavoro. I componenti sono:
- training corpus
- metodologia
- testing corpus

Se si aggiunge il ML al NLP, viene migliorato drasticamente il suo funzionamento, ma rimane legato ai dati di partenza. Per l'addestramento esistono due tipi di algoritmi:

Non supervisionato: vengono utilizzati dei corpus non annotati per creare modelli. Usati per compiti come il ranking dei dati in base a qualche funzione o il clustering in base a similitudini (*raw corpora*)

Supervisionato: si basa su un training "*annotato a mano*" tramite linguaggi di mark-up (*XML*, ...). Sono adatti per le classificazioni come, per esempio, trovare le corrette classi di appartenenza (*machine readable*)

L'annotazione può essere fatta:
- manualmente, ma è molto lenta e costosa, a volte incoerente
- semi-automatica, tramite *parsers* (corretti al 96%) o *taggers (corretti al 98%)*

L'annotazione segue un processo di sviluppo di annotazione denominato *M.A.T.T.E.R.*:
- **M**odel, descrizione del fenomeno linguistico
- **A**nnotate, annotazione in base alle features
- **T**raining, addestramento di un algoritmo su un corpus
- **T**esting, viene testato l'algoritmo su un altro corpus
- **E**valuate, viene valutato l'esito del testing
- **R**evise, viene eventualmente revisionato l'algoritmo

I testi possono essere annotati morfo-sintatticamente tramite il *PoS (Part of Speech) tagging*. Il procedimento del PoS tagging genera delle coppie di valori dove vengono associate le parole e le parti del discorso corrispondenti, ad esempio: <*cane*, *Nome*> <*mangia*, *Verbo*>

L'annotazione *BIO2* viene utilizzata per espressioni polirematiche, espressioni che sono composte da più entità, ma valgono come un'unica ("carta di credito", "scala mobile").
L'acronimo BIO sta per: **B**egin, **I**n (parte interna), **O**ther

Le annotazioni sintattiche vengono svolte tramite i *TreeBanks*, i quali danno:
- *bracketing*, strutture a costituenti
- *relazioni grammaticali*
- *struttura predicativa*

Gli schemi di annotazione possono essere:
- *labelled bracketing* (es: *[S[NP Gianni][V legge] ...]*
- *dependency structure*

Le annotazioni semantiche vengono svolte tramite la *Named Entity Recognition*, basata sulle categorie semantiche, la quale assegna ai nomi propri le "entità" della loro categoria.
Ad esempio: ("*Luigi*", "*Person*") ("*Università di Pisa*", "*Organization*")

Linguistica Computazionale

Annotazione

I dati annotati "*a mano*" devono sempre essere valutati per determinare il grado di affidabilità. Per essere tale, l'annotazione deve essere replicata in maniera coerente da più annotatori, anche in momenti differenti.

Il grado di accordo si basa sui dati di *almeno* due annotatori e viene chiamato *interannotator agreement*. Esistono diverse misure di accordo, il più comune è chiamato *k di Cohen*, che si basa su soli due annotatori. Per 3+ annotatori avremo il *multi-k*, etc.

Il *k di Cohen* misura l'accordo al netto dell'accordo che si verificherebbe casualmente. Per poterlo calcolare si crea la cosiddetta *tabella di confusione* e, sulla diagonale alto-sx basso-dx, troveremo le volte che A_1 e A_2 sono concordi.

Tabella di confusione, esempio:

A2 \ A1	X	Y	
X	50	20	R1
Y	10	20	R2
	C1	C2	

Si calcola come: **$k = (P_A - P_E) / (1 - P_E)$**

- P_A è il totale delle volte che A_1 e A_2 concordano, diviso il numero dei casi totali ($P_A = (O_{11} + O_{22}) / N$
- P_E è il totale delle volte che vengono fatte le scelte per caso ($P_E = P_X + P_Y$ dove $P_X = P_{R1} * P_{C1}$ e $P_Y = P_{R2} * P_{C2}$)

Per un buon accordo, si dovrebbe ottenere k ≥ 0.80. Per valori inferiori, il sistema potrebbe necessitare una revisione.

il *PoS tagging* è usato per la disambiguazione morfosintattica. Il compito è quello di assegnare, ad ogni token, la categoria grammaticale adatta. Esso si complica nella specifica dei tratti morfologici

Può utilizzare la regola *Pattern-Action*:

- *<action>*: seleziona e/o rimuove i tag
- *IF:* blocca *<action>* fino alla conferma
- *<pattern>:* se verificato, esegue *<action>*

Nella fase di addestramento troviamo:

- *training set*, un corpus di testi annotati
- *feature*, le caratteristiche estratte dall'input
- *language model*: *<feature, peso>*

Negli algoritmi supervisionati, il numero di classi y di un certo input x (<x;y>, coppia del *training set*) deve essere finito, mentre gli input non sono quasi mai finiti. L'algoritmo deve quindi imparare le generalizzazioni di eventi mai osservati.

Le features sono le caratteristiche che devono essere analizzate e sono indicate come coppia <attr., valore>, dove il valore è un numero binario 0 (assente) o 1 (presente).
L'estrazione delle features è uno dei compiti più complessi, poiché potrebbe falsificare il modello che viene generato.

Le features si distinguono in:

- *locali*, estratte dal token stesso (*forma*, *lemma*, ecc)
- *contestuali*, estratte dal contesto in cui si trova il token (analisi della sua "storia")
- *globali*, più ampie di quelle contestuali ma meno usate (dominio del documento)

Linguistica Computazionale

Gold Standard

Il Gold Standard (GS) è una porzione del test corpus annotato manualmente, che rappresenta l'output di riferimento.

Basandoci sugli output attestati nel GS, abbiamo:
- *True Positive* (TP), sono gli output confermati con il GS
- *False Positive* (FP), sono gli output errati nel GS
- *False Negative* (FN), sono gli output mancati ed errati (FN = |N| - TP)

La *N-fold crossvalidation* è la metodologia di valutaione ideale per quando i dati annotati sono limitati:
1. Si divide il corpus in N parti
2. Si compiono N cicli di *training-evaluation* usando a ciclo una parte come test e le n-1 come training
3. Si effettua la media delle prestazioni

La *baseline* è il limite inferiore che ci si attende dal sistema. L'*accuracy* è invece data dal rapporto degli output corretti e la lunghezza del corpus.

Information Retrieval: ci dà delle misure riguardanti:
- *precisione*, la correttezza delle risposte sel sistema
- *richiamo*, la copertura del sistema

Dai valori ottenuti con il GS possiamo calcolare tre parametri:
- *Precision* (P) = TP / (TP+FP)
- *Recall* (R) = TP / (TP+FN)
- *F-Measure* = 2PR/(P+R)

Precision e Recall sono due valori che si penalizzano a vicenda: crescendo uno, diminuisce l'altro

Natural Language Processing

È il processo di analisi di una frase per determinare la sua struttura sintattica. Possono essere a costituenti o a dipendenze.

Vengono individuate le relazioni sintattiche tra i token della frase. Producono alberi a dipendenze, dove i *nodi* sono i token e la *radice*, mentre gli *archi* sono le relazioni tra le parole.
Si usano classificatori addestrati su delle *TreeBanks* allo scopo di predire l'azione in base alle features.

Le azioni effettuate per costruire:
- *shift*: non c'è relazione tra i due token, va avanti
- *right*: c'è relazione, il nodo sx dipende dal nodo dx
- *left*: c'è relazione, il nodo dx dipende dal nodo sx

L'algoritmo termina dopo aver analizzato tutto e comincia l'analisi in base alle features estratte.

Linguistica Computazionale

Semantic Network

E una memoria semantica, organizzata come una rete di nodi (concetti) e archi (relazioni). Si basano su una struttura tassonomica.

Un concetto viene rappresentato come l'insieme delle parole sinonimeche lo esprimono. Viene chiamato *synset*.

Un *synset* a cui appartiene un concetto X, può contenere una o più parole che lo esprimono. Una parola che può esprimere più concetti, genera ambiguità lessicale. Quest'ultima apparterrà ad n synset diversi, come il lemma "*cane*" (animale, parte di pistola, ecc)

Esistono due tipologie principali di Semantic Networks:

WordNet è un SN di tipo relazionale. Può essere usato come dizionario, accedendo per lemma, o *tesauro*, accedendo tramite l'ID univoco del synset interessato.

Sviluppato a Princeton come modello relazionale riguardo nomi, aggettivi o verbi.

Rappresenta il lessico computazionale di maggior successo, con validaizione su ampia scala di pregi e difetti delle reti semantiche. La similarità semantica si basa sulla distanza tra concetti, in base al numero di nodi o archi.

Esistono dei cloni di *WordNet* come: *MultiWordNet* e *ItalWordNet*

FrameNet è un SN di tipo *frame*. Il significato di un'entrata lessicale viene descritto tramite un *semantic frame*.

Un *semantic frame* è una rappresentazione schematica di un evento in termini dei possibili pratecipanti, detti *frame elements*:

- ognuno associato alle sue possibili realizzazioni sintattiche
- per ogni realizzazione è associata una frase del corpus annotata

I lemmi ambigui sono associati a più frame

Gli elementi del *frame element* sono:

- *Lexical Unit* (LU), coppia <*parola*, *significato*>
- *Significato*, frame evocato dalla parola
- *Frame*, schema concettuale che descrive una scena con i suoi *frame elements*, che possono essere definiti come *core* (primari) e *peripheral* (periferici) FE

Word Sense Disambiguation

Ha l'obiettivo di assegnare a ogni token il significato più appropriato in un contesto (es: *BORSA*: *borsa* oggetto, contenitore - *borsa* organizzazione)

Il funzionamento avviene con il principio bayesiano su un vettore *f* formato da *al più* 3 parole a sinistra e *al più* 3 parole a destra del token interessato. Applicando Bayes, si sceglie l'elemento con valore maggiore ottenuto (*argmax*)