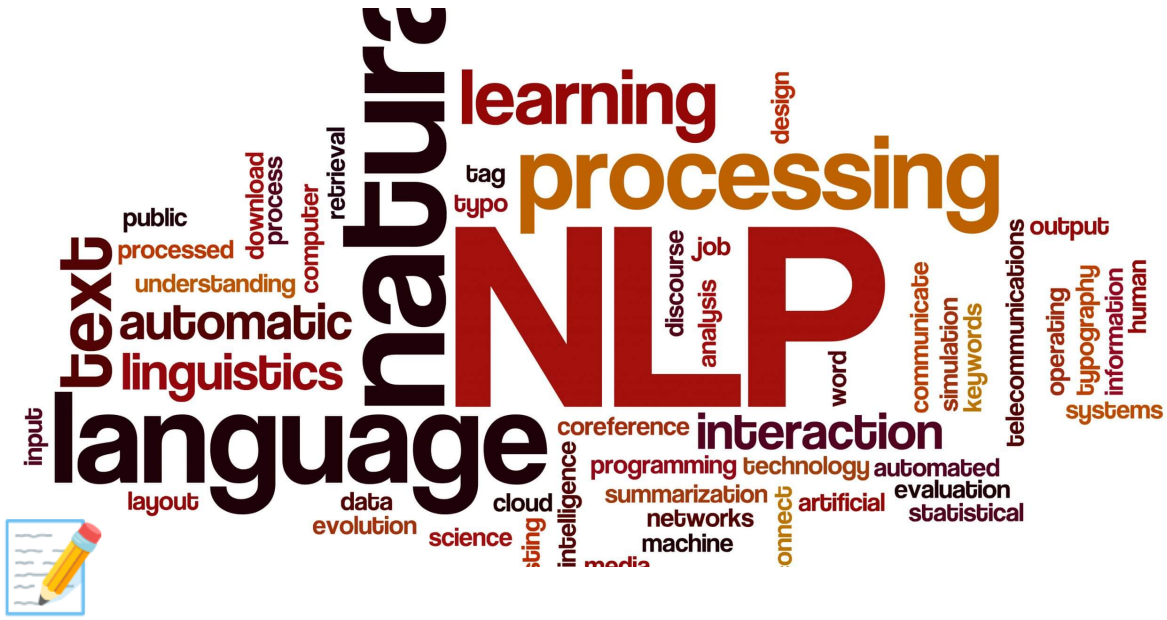


TLN [Di Caro]

💡 Questo è un riassunto / dispensa della 3° parte del professor Di Caro del corso di Tecnologie del Linguaggio Naturale (2022). Questo documento è stato definito facendo un *merge* tra le cose dette a lezione dal prof ed altri vari riassunti trovati nel repo.

💡 Preciso che potrebbero esserci errori di battitura e/o eventuali parti non ben scritte

💡 Questo documento è stato fatto con Notion, se lo usi e vuoi direttamente la pagina (per fare modifiche, ecc..), sono disponibile a passarla. Scrivimi → https://t.me/Aleeee96

▼ Panoramica sulla comprensione

Word Sense Disambiguation (WSD)

Uno strumento che svolge WSD è WordNet.

Una parola può avere tanti significati. In base al contesto disambiguiamo le varie parole. Per essere effettuata dobbiamo avere a che fare con alcuni problemi

- **specificità** granularità dei sensi troppo fine: ci sono troppi sensi per una parola. Alcuni di questi sensi potrebbero essere accorpati, in quanto molto simili tra loro. La sparsità

della disambiguazione porta a problemi nelle fasi successive.

> **Esempio** "*to play*" ha molti sensi, anche abbastanza uguali, ma li specifichiamo perché ci potrebbero servire per capire il senso nel contesto

- **copertura** alcuni percorsi della tassonomia di WordNet non sono completi e meno densi rispetto ad altri

> **Esempio** i neologismi potrebbero non essere presenti

- **soggettività** ognuno di noi può avere diverse accezzioni e assegnare significati diversi, quindi è impossibile avere una conoscenza condivisa e unica al 100 %. WN, invece fissa una conoscenza andando a perdere flessibilità.

💡 Questi problemi rendono WN molto utile, ma fragile

Word Sense Induction (WSI)

È una variante della WSD in cui non abbiamo un dizionario (tipo WordNet) che contiene tutti i sensi di una parola. Il senso, inteso come la categoria a cui appartiene una parola, viene dedotto cercando all'interno di una grande quantità di testi. In pratica viene fatto clustering sull'utilizzo di una determinata parola all'interno dei testi e in questo modo vengono dedotti i sensi legati a quella parola. Tale valutazione risulta più complessa e si basa sul metodo della **pseudo-word**

▼ Pseudo-word

💡 Il metodo delle pseudo-word quindi, permette di valutare l'efficacia di un sistema WSI

Tale metodo si basa sulla clusterizzazione e si divide in 3 fasi

1. **Merging** concatenazione di parole (random)

> **Esempio** partiamo da una parola, per esempio "*banana*" e al sistema di WSI diciamo che la parola è "*bananacane*", quindi prendo il corpus di partenza e ogni volta che compare la parola *banana* la sostituisco con *bananacane* e la stessa cosa ogni volta che trovo *cane*

2. **Clustering** applicazione del meccanismo WSI per l'identificazione di clusters (sensi identificati automaticamente), quindi vogliamo capire a cosa ci riferiamo (frutto o animale) quando compare "*canebanana*". In parole povere, dovrebbe estrarre due cluster di sensi: quello quando *canebanana* è usato secondo il senso di *cane* e quando è usato secondo il senso *banana*.
3. **Cluster-to-class evaluation** confronta e misura i cluster ottenuti automaticamente con le parole originali delle pseudo-word. Se i concetti vengono separati nettamente, il procedimento è andato a buon fine. Se il procedimento non va a buon fine troviamo dei concetti mescolati tra di loro.

WSD	WSI
inventory abbiamo un inventory (**es** Wordnet) di tutti i pissibili sensi legati ad una parola	**no inventory** i sensi li tiro fuori autimaticamente nell'utilizzo delle parole (non-supervisionato) **es** Se una parola *x* la utilizzo in certi contesti, allora li analizzo per fare una sorta di clustering di contesti così facendo induco i possibili sensi
grammard-based è la grammatica che mi orienta nella definizione dei sensi	**usage-based** è l'utilizzo delle parole che mi guida alla definizione dei sensi
human-based la creazione di una risorsa è basata sul lavoro di annotazione umano	**data-based** si usa un corpus da cui estrae il senso
evaluation qui è molto semplice, perchè ho un dataset dove per ogni parola ho associato dei sensi possibili e corretti, quindi devo solo vedere se faccio match tra i sensi del corpus e quelli trovati dal mio sistema	**evaluation** è più complicato perché una volta trovato il senso non si può confrontare con una inventory. Per cui è stato individuato il metodo della **pseudo-word**

Digressione su Lexical Semantics e risorse associate

- **Dizionari elettronici** → Wordnet, babelnet (lessema,definizione, relazione paradigmatica/sintagmatica)
- ▼ **Linguistico-cognitive** → Property norms

  che nascono da scienziati che studiano come percepiamo il mondo, come lo concettualizziamo. Si basano su norme ovvero una cosa che per noi è nella normalità. Quindi hanno condotto degli studi con molte persone e chiedevano di scrivere quello che veniva in mente sentendo una parola, per cui nelle property norms rappresentiamo cosa è un concetto e a cosa fa pensare tale concetto.

  > **Esempio**. se dico "*banana*" cosa ti viene subito in mente? E spesso verrà detto che è "*frutto*", "*gialla*", ecc..

  Le proprietà sono ordinate decrescentemente, in pratica si mettono per prima le proprietà più comunemente associate al concetto preso in considerazione **es** *banana = {frutto 43%, gialla 37%, …}*

Le property norms sono informazioni simili alla CSK (Common Sense Knowe-ledge), le caratteristiche sono

Caratteristiche

- **immediatezza** prima associazione che si fa ad un termine dato
- **percezione** come viene percepito un determinato concetto

Difetti

- **costose** la costruzione è costosa in termini di tempo e numero di persone da "*intervistare*".
- **mischiano attributi e valori** **es** *"is"* viene usato sia per la relazione di iperonimia (*turtle is a reptile*) sia per rappresentare una peculiarità (*turtle is green*).
- **sparsità** moltissime persone dicono dettagli singoli che però avranno poca importanza in quanto è detto da troppe poche persone.
- **mancano info che non hanno a che vedere con la percezione** se l'esperimento venisse effettuato su concetti astratti produrrebbe risultati poco utilizzabili.

- **Common-sense knowledge** → ConceptNet (contiene molte informazioni di natura sintagmatiche)
- **Visual Attributes** → che sono informazioni semantiche relative alla conoscenza visuale/percettiva. Una risorsa utilizzabile è ImageNet, ovvero il fratello di WordNet per le immagini. Sfruttano *ImageNet (*più di 500 concetti legati a circa 688K immagini), catturando aspetti "*osservabili*" di oggetti concreti, organizzati in una tassonomia. Non si trova, ad esempio, il concetto di "*giustizia*", ma si trova invece il concetto di "*orso*".
- **Word Embeddings** → associare un vettore numerico alle parole
- **Corpus Managers** → tool per organizzare i corpus, gestire materiale lessicale (tipo testi) e hanno meccanismi di accesso facilitati o meccanismi di creazione di corpora filtrati, mirati, dedicati (SketchEngine)

Tutte queste lexical semantics performano meglio/peggio su queste caratteristiche

- **potere espressivo**
- **scalabilità**
- **sorgente**
- **ambiguità**

Il common sense ha forte potere espressivo ma parecchia ambiguità

Come descrivere un concetto

Ci si basa su due lati/aspetti fondamentali

- **genus** ci va sempre un riferimento iperonimico per contestualizzare un'oggetto che si sta definendo **es** tavolo è un oggetto (not-living-entity...), restringendo il campo semantico. Questa caratteristica si chiama **genus** (iperonimo più comune) che rappresenta un "livello medio", non generico e non specifico, che più o meno tutti comprendiamo quando ci riferiamo al concetto.
- **differentia** tutto ciò che caratterizza quel concetto in maniera differenziale dagli altri, la discriminante. Rappresenta quella "cosa" che più caratterizza il concetto.

> **Esempio** "*banana*" è un frutto, sì, ma le caratteristiche peculiari sono la sua
> forma e il suo colore.

- **esempi** si utilizzano soprattutto per concetti particolarmente complessi.
- **relazioni semantiche** si utilizzano relazioni di tipo *part-of* **es** la gamba del tavolo

Come valutare una definizione (Onomasiologic search)

Uno dei problemi molto importanti è: data una definizione, è difficile ricondursi al concetto di partenza. Questo passaggio da definizione a concetto viene detto onomasiologic search, ed è esattamente l'opposto del dizionario (si basa su dizionari analogici) perchè si passa dalla definizione ai termini che riguardano quella definizione, e consentono si risolvere il tip-of-the-tongue problem: quando si ha una parola sulla punta della lingua, quindi si parla del contorno, per arrivare al termine voluto.

Una definizione è pessima

- quando non viene compresa
- sulla qualità del genus o sulla sua assenza
- se è presente una circularity (spiegare il concetto con il concetto stesso), può essere
  - **indiretta** dove per spiegare un concetto si usano altre parole che nella loro definizione usano il concetto che vogliamo spiegare
  - **diretta** **es** un computer è un computer

▼ Significato del "significato"

Teorie, granularità, approcci

Ci sono tantissime teorie sulla costruzione del significato da diversi punti di vista (linguistico, cognitivo, ecc.), in particolare possono essere basate su

- **primitive** per rappresentare il significato di una parola si sgretola in primitive universali che valgono per qualsiasi lingua (un modo arcaico di
  vedere la cosa)

  > **Esempio** per comprendere il significato di *scrivania* dobbiamo prima capire il concetto di tavolo e ancora prima il concetto di struttura piana con fondamenta. Il significato è dato dall'insieme di tali primitive.

- **relazioni** il significato che sta dietro una parola nasce dalla relazione con altre parole. Una parola di per sé non ha significato se non è impiegata in un contesto fatto di altre parole. Qui entra in gioco il concetto di Logic Forms e di inferenza: a determinate forme sintattiche posso associare poi un significato inferenziale, da questo deriva quest'altro.

  > **Esempio** il *telefono* si relaziona in modo che gli venga attribuito significato, ossia il telefono si usa per comunicare piuttosto che per qualcos'altro.

- **composizioni** il significato si costruisce in maniera generativa e creativa nel momento in cui si utilizzano i concetti in forme strutturali. In pratica è la composizione tra parole che dà il significato.

  > **Esempio** "*Mi muovo come un elefante*", attribuisco un modo goffo per
  > esprimere un tipo di movimento buffo che si fa.

Triangolo semiotico

Un modello del significato molto utile in questo ambito, esso rappresenta un modello del significato in cui qualsiasi concetto è rappresentato attraverso 3 componenti

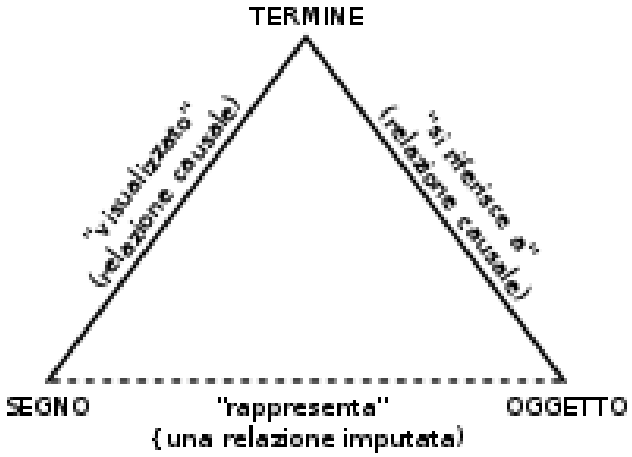
- **un concetto** (knowledge o interpretation) che è la rappresentazione che abbiamo in testa.

  > **Esempio** se parliamo di *gatti* il concetto corrisponde all'idea di gatto che abbiamo immagazzinato nel tempo all'interno della nostra mente.

- **una rappresentazione** (termini,simboli) che si basa sulle convenzioni e che ci permette di
  comunicare il concetto. Al concetto viene associato un simbolo, cioè la parola che rappresenta il concetto.

  > **Esempio**, la parola *gatto* all'interno di una conversazione ci permette di comunicare il concetto che abbiamo in testa.

- **un referente** (istanze reali) ossia l'elemento reale nel mondo, **es** il gatto vero e proprio.



Ci sono legami tra i vertici

- una **rappresentazione** simbolizza un **concetto**
- un **concetto** si riferisce ad un **referente**
- una **rappresentazione** sta per (stands for) un **referente**

Altri aspetti interessanti del triangolo sono

- Per comunicare abbiamo bisogno della rappresentazione, ma non del **referente**
- La rappresentazione è l'unico punto da cui può partire un algoritmo di IA/NLP perchè partendo da un testo avremmo solo simboli, e da questo vertice si cerca di andare verso gli altri due ovvero
  - al concetto, dando il significato della parola **es** attraverso qualche rappresentazione semantica
  - al referente, **es** attraverso delle immagini
- Ci permette di concettualizzare il lessico e i concetti per via delle connotazioni semantiche per cui ci permette di astrarre dal referente: possiamo parlare di "*gatti*"

senza essere in presenza di uno di essi. Per concetti astratti il referente permane, ma è astratto: esempio concetto di "*giustizia*" che non si può vedere o toccare con mano, ma ognuno di noi ha presente una situazione in cui è stata fatta giustizia.

> 💡 **Probabile domanda esame**
> Esistono casi in cui non c'è la concettualizzazione? Ovvero non c'è l'immagine mentale rappresentante il vertice concetto.
>
> Sì, sono le name identity (su cui si fa la name identify recognition): le entità nominali sono i luoghi fisici (Parigi, Germania), piuttosto che nomi di organizzazioni, oppure nomi di persone
> Per queste entità non esiste il vertice in alto, ma si hanno solo rappresentazione e referente.

Multilingual Word Meaning

Le lingue hanno molte cose in comune, ma anche differenze a livello sintattico/semantico. Tre aspetti fondamentali sono

- **sfida/opportunità** l'abbattimento delle barriere linguistiche è visto come una sfida ma anche come una grande opportunità. Attraverso il confronto di testi in lingue diverse è possibile comprendere sfumature di significato di altre lingue, inoltre informazioni estratte su più lingue hanno un valore più significativo. L'integrazione dei risultati estratti da testi in diverse lingue può permettermi di individuare una semantica un po' più indipendente dalla lingua e quindi è possibile diminuire la polisemia (magari in italiano una parola è molto ambigua, mentre lo stesso concetto nell'altra è meno polisemico.
- **lingue rare** che sono pseudo morte di cui non esiste un wordnet e non si vuole disperdere il loro contenuto sintattico-semantico, per creare sistemi di NLP che funzionano anche su di esse. La cosiddetta "barriera" delle lingue diverse diventa un valore aggiunto: reti neurali sfruttano lingue diverse per migliorare performance su una singola lingua.
- **concetti comunque diversi** un concetto mio è diverso dallo stesso concetto di un olandese che è molto diverso da quello di un altro paese
  con un'altra cultura.

> **Esempio** "*boh*", "*apericena*", "*mamma mia*" sono tutti termini che non hanno corrispettivi in inglese.

Granularità del significato

Ogni task di NLP ne ha una associata

- **word** complessità elevate con task word sense disambiguation
- **chunk** piccoli costrutti che vanno a formare una frase con task multi-word expression
- **sentence** singola frase con task question answering
- **discourse** insieme di frasi con task chatbot in cui ci sono scambio di frasi, ovvero si fa un discorso
- **document** insieme di paragrafi e frasi con task summarization
- **documents collection** topic modelling, data una collezione di documenti, trovare i temi della collezione

Costruzione del significato

Vediamo come si costruisce il significato utilizzando più termini insieme secondo due importanti teorie

▼ Teoria di Pustejovky

Pustejovky ha creato una teoria sulla semantica che permette di descrivere il significato che possiamo attribuire alle frasi. Nella costruzione del significato, secondo lui, la semantica interviene a diversi livelli e prevede un approccio strutturato su quattro strutture

- **argomentale** si vede in una parola che legame ha con la sintassi

  > **Esempio** il verbo può avere diversi argomenti che si relazionano ad esso

- **degli eventi** esprime il legame dell'evento con il significato. Gli eventi vengono distinti in tre classi
  - **stato** **es** "*Mary is sick*" indica uno stato in cui si trova Mary, non c'è alcun riferimento temporale sull'inizio o la fine di un evento.
  - **processo** **es** "*Mary walked*". Il verbo *walk* indica un processo
  - **transizione** **es** "*John closed the door*" indica una transizione da uno stato in cui la porta è aperta ad uno stato in cui è chiusa
- **qualia** struttura che associa specifiche proprietà ai concetti, e descrive quattro aspetti del significato di una parola, ognuno identificato come ruolo
  - **constitutive role**: relazione tra un oggetto e le parti che lo costituiscono **es** materiali, peso, parti, ecc..
  - **formal role** caratteristiche peculiari che distinguono un oggetto e lo differenziano dagli altri **es** colore, forma, dimensionalità, ecc..

    - **telic role** definiscono lo scopo e la funzione di un oggetto
    - **agentive role** legato a chi crea/genera/interagisce con l'oggetto.
- **ereditale** indica come la parola è relazione con gli altri concetti, quindi costruendo una tassonomia e gerarchia possiamo capire meglio il significato di una parola.

> **Esempio** se parliamo di "apple" ci troviamo dentro all'iperonimo "frutto"

A cosa serve generare questa teoria?

Pustejovky dimostra che nel momento in cui analizzo una frase, se per ogni parola codifico tutte queste informazioni (come spiegato sopra), allora si può costruire un modello di inferenze di ragionamento molto profondo che esplicita tutta la semantica.

> **Esempio** da una novella possiamo dire che a livello **constitutive** è narrativo, a livello **formale** e sotto forma di libro, che a livello **telico** si legge e a livello di **agente** c'è lo scrittore e il lettore

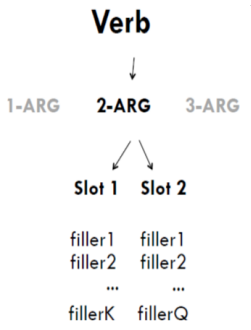
Problemi

- molto teorica e poco pratica, difficile da usare computazionalmente.
- i ruoli sono troppo specifici, non spiegati a un certo livello di dettaglio e crea ambiguità nel linguaggio

▼ Teoria di Hanks

Struttura più semplice e, soprattutto, implementabile.

Secondo Hanks il significato di una frase nasce dal verbo che per lui è la radice del significato, ad ogni verbo viene associata una valenza che indica il numero di argomenti necessari per il verbo. In base al numero di argomenti che un verbo richiede, in alcuni casi possiamo differenziarne il significato.



> **Esempio** "Io *mangio*" → valenza=1 | "*Io mangio la mela*" → valenza=2 | "*Io mangio la mela al mare*" → valenza=3

> 💡 Secondo Hanks, la prima distinzione di significato avviene a livello di valenza.

Una volta determinato il numero di argomenti avremo degli slot$(n_{argomenti} = n_{slot})$ e per ciascuno slot ci sono dei filler che possono "riempirli", questo procedimento è detto collocazione.

> **Esempio** Prendendo il verbo "*mangiare*", con valenza 2
>
> - gli slot sono il soggetto e complemento oggetto di tutte le cose possono essere prese in input a livello lessicale
> - i filler sono tutti gli elementi che possono riempire lo slot, quindi tutto ció che può mangiare o puó essere mangiato.

Hanks dice che è necessario raggruppare i vari filler secondo i semantic types che rappresentano delle generalizzazioni concettuali strutturate come una gerarchia.

> **Esempio** se siamo nei soggetti del vero "*mangiare*", ci possono essere persone, animali, o concetti astratti (**es** "*questa attività mi mangia tutto il tempo*")

Variazioni sintattiche Hanks aggiunge piccole varianti/modifiche, una è quella in cui alcune combinazioni possono essere unite perché sono varianti sintattiche con stesso significato.

> **Esempio** una frase attiva trasformata in passiva, il significato rimane lo stesso.

💡 Secondo Hanks quindi per attribuire significato ad una frase si deve prendere il verbo, capire quanti argomenti ha, recuperare tutti i filler, clusterizzare per semantic type e creare una combinazione di tutti semantic types che permettono di produrre significati diversi.

💡 Grazie a questa vesione siamo in grado di indeitificare come *"mangiare"* assuma due significati distinti

- una persona che mangia del cibo (un frutto)
- la macchinetta del caffè che mangia i soldi (spreca)

Per individuare i semantic types si può ricorrere per esempio a

- **Wordnet Super Senses** dove si ha una tassonomia in cui si hanno diversi alberi e le radici di tali alberi sono i supersensi divisi in PoS
- **CSI** che è un inventory di sensi grossolani più astratti e si hanno tutti i synset di wordnet con la categoria associata e possono essere usati come supersensi.

Problemi non è ben chiaro quali possono essere i semantic types, qual è il livello di generalizzazioni? e quindi il livello di generalizzazione dei termini dei semantic type dipende dal contesto

> **Esempio** *"lo studente va a suola"* lo studente deve rimanere studente come semantic type, oppure astratto a persona oppure astratto a entità vivente. è il contesto che mi dice quanto generalizzare, se lo studente va al supermercato allora ha più senso astrarre a persona perche l'essere studente nel contesto del supermercato non c'entra un tubo.

▼ Text Mining e applicazioni

È il processo per ricavare informazioni di alta qualità dal testo. I problemi di text mining vennero proposti per risolvere il problema del question answering. L'idea è che un utente umano possa fare delle domande a cui il calcolatore deve rispondere in base alle informazioni che trova in un testo. Le domande riguardano fatti, definizioni ecc. e questa è un area che comprende diversi task come clustering, categorization/classification, segmentation, summarization,
information, ecc. Vi sono due approcci

▼ Linguistica computazionale (top-down)

regole → dati

vecchio approccio, guidato dalla grammatica, da regole codificate ad-hoc, utilizzando formalismi logici di tipo rule-based (**es** Prolog, CLIPS)

▼ Statistica (bottom-up)

dati → regole

con l'avvento del data mining, dal testo estraiamo delle regole attraverso la statistica

Rappresentazione vettoriale (Vector space model)

Dal punto di vista statistico le parole sono dei token (sequenze di caratteri cotigui) e un testo è un insieme di token che hanno una certa frequenza, e in base a questa assunzione, un documento può essere visto come un vettore di coppie $(parole, \#occorrenze)$ detto **Vector Space Model**.

E se da un testo creo un vettore, da una serie di testi si crea una matrice, questo per poter rendere i testi computazionabili dai PC. Confrontando i vettori nella matrice si può capire quanto siano simili e quindi capire quanto i due testi siano semanticamente simili: per farlo si utilizza la

▼ Cosine Similarity

$$similarity(A,B) = \frac{A \cdot B}{\|A\| \times \|B\|} = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \times \sqrt{\sum_{i=1}^{n} B_i^2}}$$

coseno dell'algolo compreso dal prodotto dei due vettori in questione diviso il prodotto delle loro norme

> 💡 La norma si fa perchè se ho un testo di 10 parole e uno da 1000 se non nornalizzo mi crea un bias che non mi permette di comparare in maniera consistente i due vettori

La similarità del coseno permette di memorizzare la similitudine tra due vettori calcolando il coseno tra di loro. Per definizione di coseno, dati due vettori, si otterrà sempre un vettore di similitudine compreso tra -1 e +1 ma poiché le frequenze dei termini sono sempre valori positivi si otterranno valori che vanno da 0 a 1 dove **1** indica che le parole contenute nei due testi sono le stesse, ma non necessariamente nello stesso ordine. **0** indica che non c'è alcuna parola che appare in entrambi.

I metodi statistici applicati ai documenti di testo si concentrano essenzialmente su due tipi di informazioni statistiche

▼ Frequenza di una parola in un testo

dove abbiamo

- **TF (Term Frequency)** $\frac{frequenza\ token}{\#token\ documento}$

  il numero di occorrenze di un token diviso il numero di token (non è il numero di occorrenze, ma il numero di occorrenze già normalizzato, ossia una percentuale di quanto quel termine occorre all'interno del testo)

- **IDF** (**Inverse Document Frequency)** $\log(\frac{nd}{ndt})$

dove ndt numero di documenti nella collezione in cui compare quel singolo termine. nd numero documenti totali

> **Esempio** Ho 10 documenti, e utilizzo la parola *gatto* in 4 documenti quindi avrò $\frac{10}{4}$ quindi un valore più grande di 1 e quindi il log di un valore più grande di 1 è positivo.

Se $ndt = nd$ (**es** gli articoli, quindi le function words) allora si tratta di termini che occorrono dappertutto, ma non sono rilevanti, per cui il rapporto restituirà **1** e mettendo tutto nel logaritmo si ottiene $\log(1) = 0$, andando così ad esprimere che il termine t è poco rilevante.

▼ Co-occorrenza di due parole in un testo

La co-occorrenza di due parole in un testo indica la similarità tra due parole assumendo che due parole con significato simile siano presenti negli stessi contesti.

> **Esempio** *"gatto"* e *"cane"* che sono due parole molto differenti, ma che compaiono molto spesso vicine nei testi. Si suddivide il testo in paragrafi e si conta quante volte 2 parole sono nello stesso paragrafo, ovvero co-occorrono nella stessa sezione.
> *cane* e *gatto* in 5 paragrafi su 27 $\frac{5}{27}$

Questo permette di dire che se *cat* e *dog* hanno valori di concorrenza alti, vuol dire che compaiono negli stessi contesti e quindi condividono contenuto semantico (non sono proprio distanti) e questo mi permette di calcolare degli score di similarità. Quindi se il valore è molto alto si parla di vicinanza semantica (detta **distributional hypotesis**)

Applicazioni del Text Mining

Di seguito si trattano alcune delle applicazioni del text mining, sfruttando gli elementi citati fino ad ora

▼ Tag cloud

inserisce le parole di un testo all'interno di una nuvola da cui è
facile ricavare il topic del documento

tag flakes è un particolare tipo di tag cloud in cui viene fatto un ordinamento in base alla frequenza e successivamente si separano le dimensioni su una base semantica. Ciò permette di

- organizzare in le tag cloud in base al significato

- estrarre in maniera automatica una gerarchia di termini in cui sia rappresentata la reciproca similarità, in questo modo si individuano diversi topic che hanno al loro interno parole semanticamente correlate.

▼ Document clustering

mentre nel data mining si applicava il text clustering, invece del text mining parliamo di document clustering che è un meccanismo per raggruppare per similarità diversi documenti e rappresentarli nelle loro dimensioni su un piano (apprendimento non-supervisionato). Quindi ho dei documenti e voglio separarli in cluster affini e coerenti, possibilmente separati bene tra di loro.

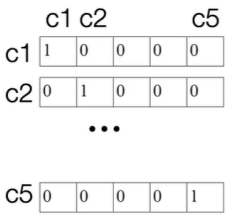
> 💡 In alcuni casi le tecniche di clustering vengono utilizzate in cascata (pipeline), ad esempio potremmo effettuare il clustering e successivamente creare una tag cloud per ogni cluster.

▼ Document categorization/classification

qui i cluster ce li abbiamo già e l'obiettivo è associare ad un documento una certa etichetta di una tassonomia (apprendimento supervisionato).

Come associare documenti di testi ad una tassonomia? Usiamo l'approccio CP/CV: ad ogni nodo, con la sua etichetta, viene associato un vettore numerico e vengono fatte due cose

- **inizializzazione** creiamo una matrice dove in ogni vettore abbiamo tutti 0, tranne nella cella che rappresenta il concetto stesso (**es** c1: *World*, c2: *America*, ecc..)



- **propagazione** essendo che i nodi hanno un intorno topologico, propaghiamo i suoi valori verso altre dimensioni vicine (dall'alto verso il basso, dal basso verso l'alto). Viene utilizzato un fattore α (detto **fattore di propagazione**) per decidere quanto trasferire da una dimensione all'altra.

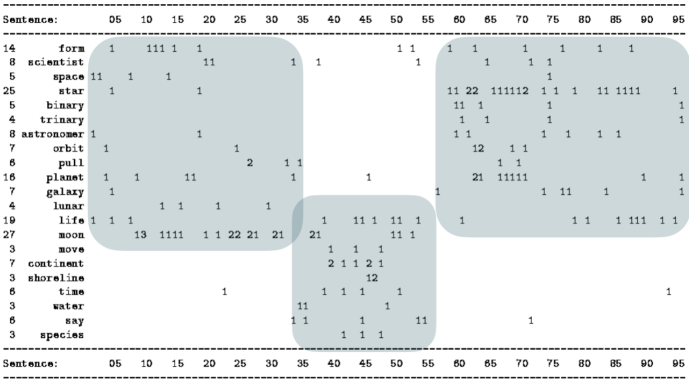
> **Esempio** I concetti *Asia-Mondo* si trasferiscono nella gerarchia$\alpha - percentuale$ informazioni

> 💡 la logica e il risultato finale è che Asia è il pezzo di mondo piú grande e viene propagata piú informazione, anche un bel pó per Africa, mentre il resto è tutto meno legato al nodo World

▼ Document segmentation

Individuare elementi diversi all'interno del discorso: si parte in **input** da un singolo testo per individuare in **output** cambi di discorso nel testo. Utile per vedere l'evoluzione di un tema o delle relazioni.

Il metodo più famoso è quello del **text tiling**



dove sull'asse x c'è il numero della frase del testo, sulle y ci sono i termini utilizzati nel testo, e all'interno della matrice ci sono le frequenze di utilizzo dei termini (**es** "*moon*" compare 13 volte nella sentence n.10) visivamente quindi si notano cluster che detonato che nelle prime frasi si parla di un certo argomento e poi di un altro (**cambio tematico**)

Come funziona?

Si basa su 4 punti

- separazione del testo in finestre di lunghezza fissa (all'inizio randomiche)
- dalle finestre identificate prima sia calcola della coesione intra-gruppo, che misura quanto sono legate le parole all'interno di una finestra (una misura potrebbe essere la co-occorrenza)
- a questo punto si cercano i **break point**, ossia quei punti in cui la coesione intra-gruppo ha un crollo perchè è molto probabile che in tali punti finisca una sezione del documento relativa ad un argomento e ne cominci una nuova.
- questo ci permette di abbandonare la lunghezza fissa delle finestre e andare ad individuare delle finestre di grandezze diverse

> 💡 Tutto questo lo fa in maniera iterativa, perché nel momento in cui all'iterazione 1 posiziona le finestre poi devo ricalcolare la coesione intragruppo (tipo k-means)

▼ Document summarization

Consiste nella riduzione del testo mantenendo il contenuto più informativo, può essere

- **estrattivo** estrazione di frasi contententi le parole più frequenti e dominanti nel testo (non genero ma snellisco solo il testo di partenza)
- **astrattivo** testi codificati in una sorta di modello (neurale), tale codifica viene data in pasto al decoder che genera frasi nuove. Metodo molto più complesso (qui genero un nuovo testo)

Valutazione

ROUGE [recall] meccanismo di valutazione basato su sovrapposizione di parole tra il documento generato automaticamente e il sommario generato da una persona (gold summary) tramite string matching

▼ Orienteering browsing

Task creato per aiutare le persone a orientarsi in una sorgente di informazione. Il problema è che non sempre le persone sanno come cercare le informazioni. Lo scopo è quello di sviluppare dei meccanismi basati sul concetto di orienteering per aiutare gli utenti a navigare tra i topic sfruttando le relazioni tra i documenti di testo. **es** evidenziando relazioni tra documenti (Wikipedia).

▼ Information retrieval

Recupero di documenti di interesse andando ad usare query basate su keyword o concetti. Troviamo l'idea di rete semantica che si forma quando esplicitiamo i legami fra le tre componenti fondamentali: query, documenti e concettualizzazione.

▼ Semantica documentale e visualizzazione

Analisi semantica a livello più alto possibile: siamo a livello di documenti interi, collezioni di materiale (non un singolo termine, paragrafo o documento, ma una collezione di essi). **Task**: Si ha una collezione di documenti, e consiste nel dire di che cosa parla questo insieme di documenti (rientra nel campo del text mining)

Topic Modelling

tecnica che meglio traduce il concetto di semantica documentale perchè quando ho una collezione di documenti o:

- faccio una **search** (information retrieval)
- modello un topic (argomento) in maniera automatica (topic modeling)

TM è modello statistico/probabilistico che si basa sull'utilizzo del linguaggio per trovare l'argomento di un documento. La radice informativa più importante sono le co-occorrenze. Per una macchina un **topic** è semplicemente un set di termini messi assieme, dove ogni termine ha un certo peso probabilistico. Dato in input una collezione di testi, vogliamo un insieme di insiemi di termini pesati (diversi topic).

> 💡 Il modello è non supervisionato perché non necessita di dati annotati.

Problemi

- **bassa interpretabilità** non è sempre facile interpretare l'output, basandosi sulle co-occorrenze, i termini tendono a comunicare un argomento a se stante, ma a volte non hanno tanto senso assieme
- **topic estratti non sono sempre utili** le liste di parole a volte sono delle function words (**es** articoli e preposizioni), oppure eventi anomali, ma anche numeri o punteggiatura. A volte ci sono dei fenomeni di combinazioni lessicali assolutamente accidentali (parole che occorrono in maniera sistematica, ma accidentale: non indicano una tematica specifica)
- **parametri statici** quasi sempre gli algoritmi di TM necessitano come parametro il numero di topic da tirare fuori, e questo è un limite perchè non è più non supervisionato allora.

Per risolvere questi problemi abbiamo introdotto **LSA**

▼ LSA - Latent Semantic Analysis

si basa su una fattorizzazione matriciale SVD (Singular Value Decomposition): una trasformazione di una matrice da spazi multi dimensionali ad altri spazi multi dimensionali ridotti (passaggio da matrice sparsa a matrice densa)

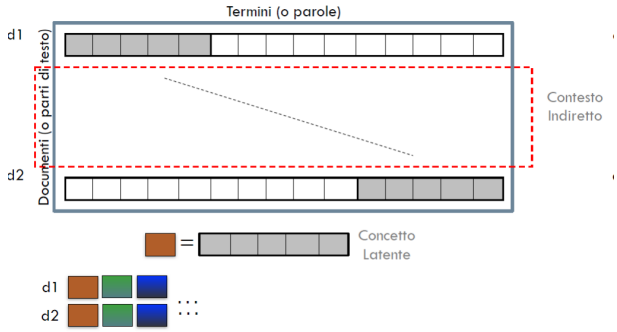
SVD data una matrice $n \cdot m$ in input, approssima quella matrice tramite la combinazione di tre matrici

1. la prima rappresenta l'informazione delle righe della matrice di partenza
2. La terza rappresenta l'informazione delle colonne

3. La seconda è una matrice diagonale che ha solo 0 e tutti 1 sulla diagonale e rappresentano delle combinazioni lineari delle righe e colonne della matrice originale. Inoltre la matrice diagonale da l'informazione su quanta entropia siamo riusciti ad indentificare

La fattorizzazione è importante perchè analizziamo la ridondanza della matrice originale, ottenendo una nuova rappresentazione vettoriale in uno spazio ridotto individuando i concetti latenti (nascosti) e concetti indiretti, cioè termini che co-occorrono in maniera molto simile

Colonne = feature, ovvero termini/parole
Righe = documenti (d_1,d_2..)



Le celle bianche indicano che quella parola non è usata in quel documento specifico, viceversa in quelle grigie si. In questo caso d_1 e d_2 non condividono alcun termine, ma se facessimo una fattorizzazione matriciale i termini che co-occorrono spesso, verrebbero identificati e fusi in un nuovo spazio vettoriale ridotto

> **Esempio** d_1 parla di cani d_2 di gatti, quindi apparentemente sono documenti diversi, ma in realtà non lo sono perchè cani e gatti sono animali dimestici, e quindi hanno cose in comune tipo la parola "*cuccia*" che è un concetto **latente** perche lega cane e gatto non in maniera esplicita

Quindi, si capisce automaticamente tramite misure di co-occorrenza che alcune parole sono diverse dal punto di vista lessicale, ma sono affini perchè co-occorrono sovente

Problemi

- **non generalizza su documenti che non ha mai visto** parte da una matrice e crea i concetti latenti, ma una volta arrivato un nuovo documento non è possibile mapparlo all'interno della matrice stessa, quindi bisogna ricostruirla da zero con la SVD
- **presenza di valori negativi** dopo SVD potremmo avere dei valori negativi difficili da interpretare (buona prassi è metterli tutti a 0, **Non-negative Matrix**

Factorization)

Per risolvere tali problemi sono state proposte diverse varianti

▼ Latent Dirichlet Allocation (LDA)

versione probabilistica di LSA che sfrutta la statistica bayesiana e si basa sul concetto che un documento è un mix di topic e ogni parola ha una certa probabilità che compaia in ogni singolo topic.

> 💡 è più potente di LSA perche se ho una probabilità che un termine compaia in un topic allora ho un task generativo che mi permette di inferire dei topic anche su un documento nuovo in input

▼ Dynamic Topic Modeling

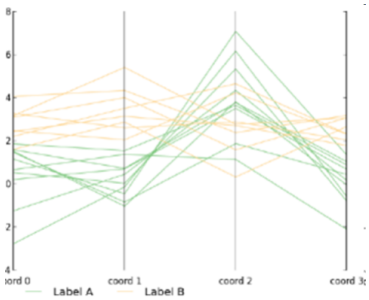
versione temporale: studia come i temi cambiano con il passare del tempo basandosi su collezioni di documenti. Nato per vedere quali temi scompaiono e quali acquisiscono importanza

Text Visualisation

Un problema comune al text mining e dalla semantica documentale è la visualizzazione. Il problema è rappresentare uno spazio n-dimensionale (quello dei termini) in uno spazio bidimensionale. Si utilizzano delle tecniche di text visualization
Una delle strategie è quello della fattorizzazione SVD usata da PCA, SOM, ecc..
Vi sono poi degli approcci grafici, tra cui

▼ Parallel coordinates

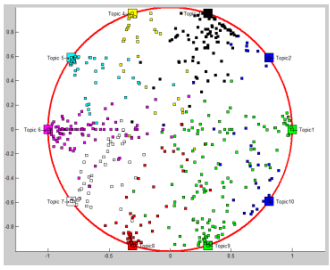
utilizzate quando la dimensionalitá va da 3 a 10. Queste dimensioni vengono proiettate come coordinate parallele, un valore è rappresentato da una *spezzata* che collega i valori sulle 4 dimensioni. Otteniamo spezzate che si sovrappongono e facendo clustering visivo tra queste righe posso individuare pattern visivi che si comportano in maniera diversa da altri pattern



▼ RadViz

permette di visualizzare massimo 10/20 dimensioni organizzate su circonferenza. Ogni termine rappresenta un punto di attrazione nella circonferenza (i punti sulla circonferenza sono i topic). I vari documenti si

dispongono a una distanza dai topic proporzionale al numero di occorrenze di quel termine.

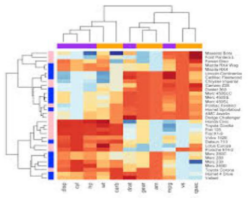


> 💡 Si può pensare che queste dimensioni siano dei magneti: piú è altro il valore per quelle coordinate e piú il punto viene attratto dalla dimensione posta sulla circonferenza.

Problema possiamo avere dei punti vicini che in realtà sono dissimili

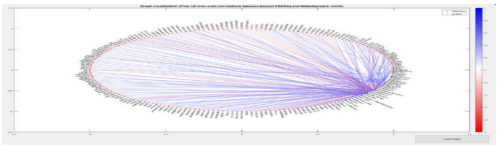
▼ HeatMap

permette di visualizzare dimensionalità molto elevate, dove i valori contenuti nella matrice sono rappresentati da colori



▼ Correlation circle

i termini sono disposti lungo la circonferenza e collegati fra loro mediante delle rette che ne indicano la correlazione. Utile per le matrici di co-occorrenza.



▼ Semantica Distribuzionale

È una rivisitazione in chiave linguistica delle tecniche di text-mining

Semantica distribuzionale trasformazione di testi in vettori e matrici.
La sua diretta evoluzione è il word embedding.

▼ Storia e citazioni a riaguardo

- Nasce tutto negli anni 50 con **Harris** e **Firth** che danno vita al concetto di distributional hypotesis
  - Harris affermava “*le parole che occorrono negli stessi contesti tendono ad avere significati simili*”

    - Firth similmente diceva *“una parola è caratterizzata da quelle che la accompagnano”*
- Negli anni ’80 **Furnas** riprende questo concetto andando a dire che “P*rendere più keyword insieme (quindi prima una parola chiave, poi un’altra e così via) serve a specificare in maniera più dettagliata l’oggetto del discorso”.*
- Negli anni ’90 **Deerwester** parla di semantica latente secondo la quale i dati sono "*oscurati*" dalla scelta randomica delle parole che fanno riferimento ad un documento di un autore che magari ha usato altre parole per descrivere un concetto già descritto da altri.
- Negli anni 2000 si fornisce un modello generativo in grado di creare documenti mai visti prima. **Blei** afferma *“Il tema del documento (ciò che mi accingo a scrivere) influenza probabilisticamente la scelta delle parole”*
- Sempre in quei tempi **Turney**, dice che “*coppie di parole che co-occorrono in pattern simili tendono ad avere delle relazioni semantiche simili”* (non si parla di vicinanza semantica di una parola con un altra, ma tra coppie di parole).

Perchè le matrici?

perchè è comodo per le macchine che lavorano bene con numeri e matrici. Ed è utile anche per distribuire i pesi nella matrice.
Fondamentale è il concetto di similarità in NLP: i simboli sono slegati fra loro ma le macchine possono calcolare una similarità distribuzionale.
Si usano le matrici anche perchè la comprensione umana del linguaggio è di fatto una concetto basato sull’approssimazione.

> 💡 Una cosa che manca nelle matrici è l’ordinamento delle parole, avendo solo le frequenze

Esistono 2 correnti di pensiero/rappresentazioni tra cui le matrici si collocano

- **rappresentazione simbolica** semplice da utilizzare, ma povera di informazione.
- **rappresentazione associazionistica** proviene da studi di psicologia in cui si afferma che la rappresentazione del significato nella nostra mente è funzione di tutto il nostro vissuto.
    - concetti simili sono vicini ed esiste un centroide che è il prototipo.
    - basata su spazi concettuali: visione degli oggetti basati sulle qualities di tipo visivo e percettivo (**es** un colore è definito dalle sue componenti RGB)

Per ottenere risultati migliori si applicano delle operazioni di pre-processing

- **Normalizzazione** procedimento lessicale, sintattico o morfologico che consiste di tokenizzazione, stemming e lemmatizzazione che vanno a restringere la cardinalitá del dizionario
- **Denormalizzazione** si aumenta il carico dimensionale ma su aspetti semantici (e non aspetti morfologici/lessicali) tramite alcune tecniche
  - **named entities** con cui cerchiamo di estrarre nomi dal testo
  - **semantic roles** con cui andiamo a determinare i ruoli all'interno della frase (agent, patient, ecc..)

Uso delle matrici nella DS

Turney nel 2010 fornisce una classificazione delle matrici nell'ambito della semantica distribuzionale. Secondo lui ne esistono di tre tipi

- **term-document matrix** le classiche, ovvero quelle dove su ogni riga abbiamo un documento e su ogni colonna un termine.

  Task che posiamo applicare: similarità, clustering, classificazione, segmentazione e parzialmente question answering

- **term-context matrix** generalizzano le term document matrix: su ogni riga c'è il contesto e su ogni colonna un termine. Il contesto non deve essere necessariamente un documento, ma può essere una frase, un paragrafo…

  Task che posiamo applicare: similarità, costruzione di cluster, classificazione di parole, generazione automatica di un thesaurus, disambiguazione, etichettatura semantica dei ruoli.

- **pair-pattern matrix** il focus è su una coppia di parole che costituisce una riga, mentre come colonne ci sono i pattern che legano le due parole x e y.

  > 💡 Non si utilizzano triple di parole per due motivi
  >
  > - costo computazionale eccessivo
  > - inutile perché otterremmo matrici troppo sparse

  > **Esempio** X is solved by Y, X solves Y.
  > **Esempio pratico** riga formata da *apple:fruit* e come pattern ci sono "*isA*","*kindOf*", ecc. e quindi le celle inerenti a tali pattern avranno valori positivi

  Per pattern, si intende dei pattern lessico-sintattici in mezzo a x e y, cioè tutto ciò che lega una parola all'altra

> 💡 Si hanno delle coppie perché non si analizza la semantica distribuzionale di una parola sola, ma di piú parole.

Se prendo (x, y) e (z, w), se risultano simili tramite cosine similarity, ossia condividono uno spazio comune, allora le loro relazioni semantiche sono simili

Task che posiamo applicare: creare clustering di coppie di parole simili, dando vita a **similarità relazionali**. Si può anche fare relational search **es** **"***restituisci lista di tutte le x tali che x causa il cancro"*. Quindi non cerco una risposta in una collezione, ma cerco una relazione, a condizione
che sia la causa (o altre relazioni) di qualcos'altro. Si arriva perciò ad un livello semantico più profondo.

Similarità

Quinn affermava che la semantica distribuzionale viene detta anche semantica della similarità ed è fondamentale anche per la cognizione umana, noi umani impariamo dal passato e dalle similarità: se vedo un animale nuovo simile ad una tigre scappiamo.

Esistono diverse similaritá in ambito di NLP

- **semantic similarity** la più usata (quasi abusata) e identifica concetti
  che sono quasi sinonimi che a volte viene mal interpretata

  > **Esempio** *PC* e *stampanti* a volte visti come semanticamente simili per il contesto in cui si trovano, in realtà sono due concetti ben differenti. Mentre *bike* e *bicycle* che sono semanticamente simili

- **sematic relatedness** riguarda dei concetti che condividono delle proprietà, quindi è legato a relazioni di meronimia

  > **Esempio** *automobile* e *motore* (meronimia) sono legati da una semantic relatedness.

- **attributional similarity** simile della semantic relatedness ma più specifica, concetti con alta attributional similarity hanno degli attributi/proprietà in comune
- **taxonomical similarity** riguarda concetti che condividono degli iperonimi
- **relational similarity** ovvero similarità tra coppie di parole o meglio tra relazioni **es** *gatto:miagola* tanto quanto il *cane:abbaia*
- **semantic association** parole che co-occorrono frequentemente in grandi corpora. **es** *culla* e *bambino*

Problemi delle rappresentazioni matriciali nella DS

Esistono due principali problemi

- **non considerano l'ordine delle parole** **es** *"ho mangiato una mela"* vs *"mangiato ho mela la"* dal punto di vista numerico dei vettori sono identici
  Per mitigare il problema le matrici pair-pattern sono leggermente più sensibili all'ordine, in base a come sono scritte le coppie di parole. **es** una coppia di parole ab avrà una distribuziona diversa rispetto a ba
- **rappresentazione non composizionale** spesso le matrici sono orientate al significato di singole parole

  Per mitigare il problema l'ordine delle parole può essere preso in considerazione attraverso combinazioni vettoriali: combinare vettore *"good"* con vettore *"night"* può creare una frase di ordine corretto

Possiamo anche ricorrere alll'**arricchimento matriciale** che aggiungono informazioni alla matrice mediante delle risorse esterne esistenti, tipoo dai Knowledge Graph codificati in un DB: semantica giá codificata in termini di nodi, azioni ed etichette.

Significato filamentare fa riferimento al fatto che si riesce ad identificare un concetto a partire da un termine, andando ad usare un numero molto piccolo di feature.

> **Esempio** se ho il cocnetto *banana* non ho bisogno di chissà quali feature per identificarla

---

▼ Ontology Learning & OIE

è l'apprendimento di ontologie consiste nell'estrazione semi-automatica di concetti e relazioni rilevanti a partire da una collezione di documenti o altri insiemi di dati al fine di creare un'ontologia

Per l'apprendimento di un'ontologia solitamente si utilizzano strumenti automatici attraverso **reverse engineering:** data una conoscenza di un certo dominio con la sua rappresentazione e la sua scrittura, vogliamo tornare alla concettualizzazione di partenza (c'è una similarità con il triangolo semiotico)

La domain knowledge dipende perciò dal dominio, quindi non si codifica tutta la conoscenza, dipende dal task (commintment ontologico)

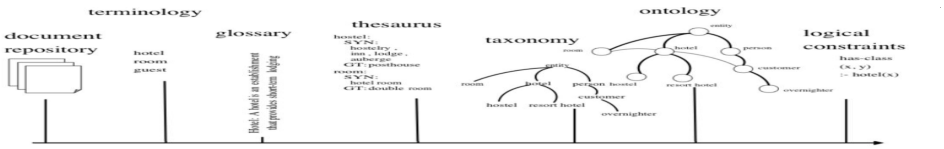
> **Esempio** se devo creare una tassonomia degli alimenti, se i miei task sono di classificazione dei prodotti in un supermercato non mi interessa di andare nel dettaglio dei singoli ingredienti nei prodotti

Differenze con altri tipi di ontologie

ontologie che fanno parte dell'ontology leanrning ma che si differenziano per certi task

- **ontology population** parte da una ontologia già fatta e si cerca in una base documentale le istanze da associare a tale ontologia (**es** tipo riempire un frame)
- **ontology-based annotation** l'attenzione è sui documenti → simile al precedente, parte da ontologia con delle foglie e cerca i termini delle foglie e li annoto con i concetti di appartenenza. In pratica etichetta parti di testo legandole a specifici concetti dell'ontologia
- **ontology enrichment** cerco di arricchire un'ontologia dal punto di vista di concetti e relazioni mancanti, lo scopo non è solo popolarla, ma capire cosa manca in quell'ontologia

Livelli di ontologie



- **documenti**
- **terminologia**
- **glossario** in più del dizionario possiede una glossa (definizioni + esempi)
- **thesaurus** tipo WordNet, abbiamo le prime relazioni
- **tassonomia** relazioni gerarchiche
- **ontologia** livello strutturato, include diversi tipi di relazioni, regole e assiomi
- **logico** regole simbolico-formali dove si specificano delle regole di inferenza

Cosa vogliamo estrarre?

Questi task fanno ancora riferimento ai livelli di ontologie, ce ne sono diversi e di diversa natura/complessità

- **term extraction** se voglio estrarre solo nomi per concetti e relazioni (**es** tag cloud)
- **synonym extraction** si vogliono estrarre sinonimi
- **concept extraction** può essere
  - *intensionale* (concetto → glossa), si fa un gloss learing, elenco di parole che identificano un argomento

- *estensionale* (glossa → concetto), creare le istanze possibili di un determinato concetto

- **concept hierarchies induction** si parte dai concetti preesistenti e si vuole indurre il rapporto tassonomico tra di essi
- **relation extraction** trovare relazioni semantiche tra concetti esistenti
- **population** estrazione di specifiche info (**es** prezzo) collegandole a delle name identity linking (**es** Apple come azienda).

Come vogliamo estrarre?

si possono usare tre paradigmi

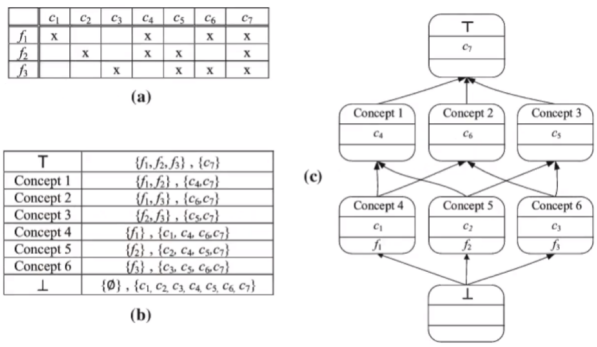
▼ NLP classico

basato

- sul ruolo dei PoS in quanto indicativi per la semantica **es** i verbi tipicamente sono azioni/eventi, gli aggettvi sono attributi, ecc.
- dopo si eseguono operazioni di preprocessing **es** tokenizzazione, stemming/lemmatizzazione
- dopo abbiamo una fase di analisi sintattica con un sistema a regole basato su alberi di parsing **es** se ho un determinato verbo, prendo il complemento oggetto del verbo e lo associo a un concetto nell'ontologia
- dopo si effettua un'analisi semantica tramite vector space model
- infine, possiamo utilizzare risorse linguistiche esterne a supporto **es** WordNet,BabelNet e FrameNet.

▼ FCA (Formal Concept Analysis)

metodo che proviene dall'ambito matematico per fare estrazione di tassonomie. Usato a partire da rappresentazioni vettoriali per poi arrivare ad una rappresentazione gerarchica.



- (a) rappresenta la *matrice concetto-feature*, rappresentiamo i concetti attraverso un insieme di features
- (b) si chiama *contesto formale* abbiamo la lista dei concetti e due simboli $\top$ è la testa, ovvero dove ho tutte le feature, $\bot$ che non ne contiene alcuna, e poi degli insiemi di feature e concetti
- (c) abbiamo la rappresentazione a lattice dove in cima abbiamo c_7 che ha tutte le feature e più andiamo verso il basso rappresenteremo concetti con meno features. Da questo si può costruire una tassonomia infatti, esistono dei concetti cha hanno più proprietà di altri → più generici; meno proprietà → più specifici.

> **Esempio** *mela* e *pera* hanno features in comune, ma ad un certo punto avranno delle features diverse l'una per l'altra (la *forma* tipo), quindi si sotto specificano

▼ ML (Machine Learning)

Possiamo utilizzare approcci

- **supervisionati** come classificatori bayesiani, alberi di decisioni, SVM
- **non supervisionati** come il clustering, che può essere divisivo o agglomerativo.

Open Information Extraction (OIE)

É una specie di Ontology Learning. Nasce dalla necessità di estrarre grandi quantità di informazioni da grandi corpora in maniera veloce, approssimativa, ma efficiente

Lo scopo è focalizzarsi sulla granularitá delle frasi e tirando fuori **triplette** (Hanks theory) della forma *(arg1, verbal phrase, arg2)* su cui è possibile effettuare un'analisi più strutturata piuttosto che una semplice analisi sul testo permettendo di rispondere a domande tipo *"chi ha fatto cosa?*

Esempio *"Dante wrote the Divine Comedy"* → (*Dante*, *wrote*, *Divine Comedy*).

Nato per rendere scalabili i sistemi di information extraction, perché se io ho un meccanismo di question answering che mi strutturi un testo, riesco ad interrogare una base documentale di qualsiasi tipo per avere un'informazione del tipo *soggetto-verbo-oggetto*

Problemi non esiste un approccio rigoroso e unico: esistono troppi sistemi diversi, rendendo così difficile da valutare l'estrazione perché non c'è uno standard per valutare le qualità delle triplette.

▼ Knowledge Graph

Si cerca di unire il mondo NLP con le reti semantiche

Caratteristiche

- **legami (relazioni)** sono molto importanti, si ha a che fare con termini che assumono importanza quando messi in relazione (WordNet può essere visto come un esempio di KG)
- **altamente scalabili** usati nei social network **es** relazioni di amicizia
- **operazioni poco costose** mettendoli a confronto con dati SQL vediamo come sia molto più semplice collegare i dati anche perchè non si hanno dei record come nei DB classici.
- **livello computazionale** ottimizza l'accesso alle informazioni.

Non sempre è facile far diventare un grafo un qualcosa che non lo è: serve quindi trasformare i dati per poterci applicare gli algoritmi tipici dei grafi (preprocessing dei dati).

Modello

Prevede due entità: **nodo** e **relazione,** nel caso di **Neo4J** esistono delle proprietà che possono essere associate a dei nodi

- direzione relazione
- chiave-valore
- label e type

KG e NLP

	Vantaggi	Svantaggi
KG	l'informazione codificata è già sturrurata e si spiega da sola, i nodi sono dei concetti chiari e le relazioni sono tipizzate quindi la struttura è autoesplicativa	non tutta la conoscenza ha la forma di un grafo, spesso bisogna ricorrere ad una trasformazione
KG in NLP	approcci neurali recenti rappresentano lo stato dell'arte in termini di risultati in molteplici task **es** BERT	a volte servono in input di corpus in grandi quantità e qualità e a volte la logica interna dei modelli addestrati non è direttamente consultabile ed interpretabile (difficile interpretabilità)

Quando un KG è limitato, entra in supporto NLP per estrarre ciò che non è stato ancora codificato, ma non sempre serve fare NLP avanzato o almeno quando della conoscenza è proiettata nella KB/KG.

Cosa ci si puó fare con i KG?

con i KG posso farci determinate analisi

- **centralità** cercano di analizzare la topologia dei grafi per capire l'importanza di alcuni nodi rispetto ad altri nodi **es** pageRank
- **similarità** calcolano similarità tra nodi diversi
- **percorsi** lunghezza, ottimizzazione funzioni **es** cammini min/max)
- **embeddings** cercano di associare info vettoriale ad ogni nodo
- **predizione** permette di lavorare su vari task come *community*, *link-prediction* (avendo tutti i nodi, si possono stimare eventuali link mancanti).

> **Esempio** concetto di *mela* legato a diversi concetti nel grafo, ho un nodo *mango* e mi manca il link con *"è compostabile"* siccome il contesto di *mango* è molto simile a quello di *mela*, si può predire il link tra mango e *"è compostabile"*

Esistono due approcci metodologici diversi

- **data driven** analizzo i dati e vedo cosa tiro fuori per poi applicare algoritmi, si tratta di un approccio esplorativo guidato dai dati
- **goal oriented** si sa qual è l'obiettivo da raggiungere e a ritroso si cerca di capire quali info del grafo servono e quali algoritmi applicare. Riservato alle aziende **es** minimizzare il tempo, le risorse

Vengono usati anche per altri tasks

- **sense disambiguation** due parole completamente diverse avranno due grafi e sottografi distinti, è d'aiuto per capire quale senso applicare ad una parola
- **question answering** info di wikipedia raccolte e organizzate con relazioni tipiche dei grafi. **es** si può rispondere a domande tipo *"chi ha fatto cosa e perchè?*

Google KG: un esempio reale

usato nel motore di ricerca con dimensioni immense

- 600 milioni di nodi
- 187 miliardi di fatti e relazioni.

ha tanti scopi

- trovare le giuste info (disambiguazione, Information Retrival)

- summarization
- anticipazione di nuove domande

▼ NLP e Machine/Deep Learning

Il ML e il DL viene utilizzato in alcuni aspetti del NLP

Si adotta un approccio a **pipeline:** abbiamo dei dati e le loro features → le diamo in input a un modello → che utilizziamo per classificare e valutare le performance

Machine learning per NLP

- naive bayes
- support vector machine
- hidden markov model
- conditional random fields

Deep Learning per NLP

▼ Recurrent Neural Networks (RNN)

É una rete che piano piano va
avanti nella lettura e cerca di ricordare la traccia di quello che si è detto per dare un output.

Problema è che non scalano molto bene perché se il testo è molto lungo,
dal punto di vista computazionale è molto difficile utilizzarle (esplode il problema)

Un passo evolutivo delle RNN sono state le LSTM

▼ Long Short Term Memory (LSTM)

Si basano sul concetto che mentre leggono, alcune cose sono piú rilevanti di altre, e quindi si possono dimenticare.
Risolve il problema delle RNN perché tante cose che si sono dette, le perde e quindi riesce a gestire sequenze lunghe di testo e tanti dati in input e alla fine si ottiene un modello che ricorda sia **long term** (cose che si sono lette molto tempo prima), ma anche le **short term** (anche quelle che si sono lette poco prima)

Spesso sono migliori le **GRU (Gated Recurrent Unit)** è un approccio piú semplice delle LSTM e seppur semplificando molto l'architettura, su pochi dati funzionano al pari del LSTM, mentre se sono molti dati sono superiori.

▼ Convolutional Neural Networks (CNN)

Si parte dal numero di filtri (finestre) che scorrono sopra la matrice e fanno una sorta di parsing della matrice di partenza, ossia ogni volta mi danno una proiezione della matrice che è una sottomatrice. Ogni sottomatrice viene traslata su un vettore in maniera iterativa, ossia vado a fare tramite dei meccanismi chiamati di **pooling**, delle somme che contribuiscono ai
valori medi su tutta la sottomatrice. Vado quindi a creare un vettore finale dove ho ogni singola cella che collassa un pó di tutta l'informazione che arriva da una sottosegmentazione dell'input con il meccanismo di pooling, e con questo vettore si arriva a un modello classico, dove da un set di features vado a decidere qual'è la classe finale.

▼ Transformer

sono lo stato dell'arte e si basato su un'architettura

- **encoder** un set di features viene mappato su una dimensione molto piú piccola, simile alla latent semantics che cerca di comprimere una dimensione ridotta
- **decoder** presa tale dimensione ridotta, cerca di ridescriverla e riproiettarla in uno spazio multidimensionale.

L'**attention** è un meccanismo che permette alle reti neurali di focalizzarsi piú su alcune parole che su altre (danno peso alle singole parole rispetto al task). **es** BERT si basa su tale meccanismo
Questo ha creato un sistema per apprendere in maniera completamente non supervisionata, senza utilizzare etichette, ma solo il testo
strutturata.

▼ Auto Encoders

collassiamo l'input su pochi neuroni e cerchiamo di ricostruire l'originale. Nel momento in cui la rete si stabilizza (buon rapporto tra input e output ricostruito), prendiamo i livelli di mezzo (hidden nodes) che sono piccoli e questi sono i concetti latenti del LSA (ossia una bassa dimensionalitá che spiega il nostro input).

Problemi aperti

- **overfitting** accuratezza sul training set ma schifo sul test set
- **few-shot learning** learning con pochissimi dati di esempio
- **domain adaptation** adattare degli embeddings e darli in input ad un altra rete che fa ancora dei cicli di learning su alcuni dati specifici su un dominio
- **interpretabilità** le reti neurali sono delle black box e non si sa come interpretare i parametri
- **common sense** informazioni che spesso non sono nell'imput ma appunto ci va del common sense esterno per poter appredenre l'input

- **costo** nell’uso di reti neurali sia fisici che energetici
- **sviluppo** di reti possibilmente inseribili su dispositivi mobile