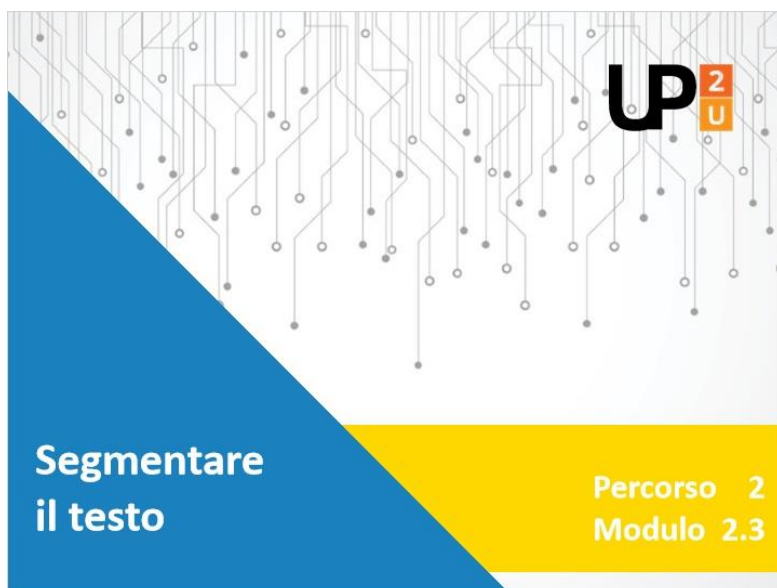


## ***1.1 Segmentare il testo***



## ***1.2 Un'elaborazione estremamente complessa***

L'elaborazione del linguaggio naturale (in inglese, NLP: Natural Language Processing) comprende quell'insieme di metodi e strumenti con cui un computer elabora le produzioni linguistiche scritte o parlate.

Uno dei principali obiettivi è la comprensione del significato, che, a sua volta, permette, per esempio, la traduzione automatica o l'interazione con una macchina attraverso comandi vocali.

Noi esseri umani spesso comprendiamo il significato senza uno sforzo consapevole. Non è così per una macchina. A volte è difficile dare un significato univoco a una parola senza sfruttare conoscenze sullo specifico contesto del discorso e sul mondo in generale. Conoscenze di cui la macchina spesso non dispone o che non è in grado di utilizzare.

Ed è per questo, che per arrivare al significato di un testo scritto o di un discorso, la prima cosa che un computer deve fare è l'analisi a livello del lessico...

## ***1.3 L'analisi lessicale***

L'analisi lessicale è quel processo che trasforma un testo in una sequenza di parole. Anzi, di token.

Un “token” è una sequenza di caratteri che rappresenta l'unità di base all'interno di un testo, il punto di partenza per l'analisi della struttura e del significato del testo stesso.

Il termine inglese “token”, di difficile traduzione, è usato in senso tecnico, al posto del più familiare termine “parola”, per denotare ciascuna occorrenza (cioè ciascuna ripetizione) di una certa parola nel testo. Il termine “parola” si usa allora per denotare il modello comune a più token.

Inoltre, nel contare i token in un testo, si considerano anche sequenze di caratteri - come numeri, elementi di punteggiatura, simboli - che normalmente non consideriamo parole.

Così in questa celebre poesia di Ungaretti a due parole delimitate da spazi corrispondono quattro token.

E nella celebre formula di Einstein i token sono tre.

Nell’elaborazione automatica dei testi, la prima cosa da fare è, quindi, la tokenizzazione, cioè segmentazione in token.

Che sembra facile, ma non lo è affatto...

## ***1.4 La tokenizzazione***

Per tokenizzare un testo bisogna individuare cosa separa un token e l’altro.

In lingue come l’italiano e l’inglese, per separare le parole usiamo uno spazio tipografico.

Lo spazio non è un “vuoto”: è un carattere come tutti gli altri a cui nei testi elettronici corrisponde un preciso codice Unicode o ASCII.

Per rendercene conto, basta un salto nel passato. In questa epigrafe del sesto secolo, quando lo spazio non era ancora stato inventato, si usava come separatore un punto al centro tra le parole!

Il fatto che lo spazio sia un carattere è importante, perché solo così un computer può leggerlo facilmente.

Ma interpretare uno spazio non è così semplice, perché ci sono token composti da più di una parola. “Los Angeles”, per esempio, può essere trattato come un unico token.

Così come le date... i prezzi... e alcune espressioni particolari che separate non avrebbero senso.

## ***1.5 L'ambiguità della punteggiatura***

A complicare ulteriormente le cose ci sono token che non sono separati da spazi, come abbiamo visto con “M’illumino d’immenso”.

Infatti, come è intuibile, anche i segni della punteggiatura fungono da separatori.

Prendiamo questa frase:

Qui il punto indica la fine della frase e quindi delimita il token “importante”. Non solo: è un token esso stesso. Per cui la tokenizzazione dà questo risultato:

Ma, come per lo spazio, nella punteggiatura anche il punto non è sempre un separatore. Infatti può far parte di un’abbreviazione... di un acronimo... di un indirizzo web... di una data... o di un numero decimale nella notazione anglosassone.

## ***1.6 Come tokenizzare una frase***

Se fossimo noi a dover tokenizzare un testo, probabilmente dopo un po’ di pratica ce la caveremmo piuttosto bene, perché siamo in grado di capire il contesto e usare il buon senso.

Non ci capiterà mai di scambiare la città de L’Aquila (che è costituita da un solo token) con un rapace, “l’aquila”, in cui i token sono due.

E non ci sbaglieremmo nemmeno se l’Aquila-città fosse scritto per errore senza la maiuscola.

Ma un computer come fa?

Nell’elaborazione del linguaggio naturale sono stato sviluppati due approcci completamente diversi al problema.

Vediamo quali...

## ***1.7 Tokenizzare con regole***

Il primo approccio alla tokenizzazione consiste nello scrivere un programma che contiene le regole da seguire.

Queste regole devono comprendere:

- i criteri generali sull’uso come separatori dello spazio, del punto, della virgola e degli altri segni di interpunzione;
- tutti i modi conosciuti per scrivere ore, date, prezzi e altro;
- l’elenco delle città composte da più parole;
- l’elenco delle abbreviazioni in uso;
- e tutti gli altri possibili casi particolari.

Naturalmente algoritmi del genere non danno risultati perfetti al primo colpo.

Dovremo sicuramente intervenire a più riprese, aggiungendo nuove regole tutte le volte in cui il

risultato non è quello atteso.

Fortunatamente c'è un secondo approccio, più adattabile, che fa largo uso dell'“intelligenza” dei computer.

### ***1.8 Tokenizzare con esempi***

Il secondo approccio consiste nel creare un programma che apprendere a tokenizzare un testo partendo da un insieme di esempi concreti chiamato “training set” (traducibile con “insieme di addestramento”). Utilizzando algoritmi di machine learning, il computer si comporta come se apprendesse da solo le regole di tokenizzazione, inducendole dagli esempi, anche se in realtà costruisce e utilizza strutture di conoscenza abbastanza diverse.

Se però gli esempi non sono ben scelti, il risultato è scadente.

Vuoi un esempio?...

### ***1.9 L'importanza dell'addestramento***

Proviamo a tokenizzare la frase “M'illumino d'immenso” con il dimostratore online “TreebankWordTokenizer”, che usa un algoritmo implementato con la libreria Python NLTK (Natural Language ToolKit), uno dei più popolari strumenti per il Natural Language Processing.

A questo indirizzo troviamo un sito che permette di sperimentare diversi tokenizzatori.

Con “TreebankWordTokenizer” il risultato è questo:

Come vedi non riesce a dare risultati corretti, anche se è stato addestrato con un grande corpus di testi annotati manualmente. Ma erano testi inglesi, non italiani!

C'è da aggiungere che, questi algoritmi a volte possono imparare in modo autonomo, dopo una fase di “addestramento” in cui, per esempio, un operatore umano segnala gli errori commessi. Oppure fornisce un “training set” più ampio.

In tutti i casi, come si è visto, gli algoritmi di tokenizzazione hanno bisogno di un processo di verifica e affinamento prima di essere affidabili.

### ***1.10 Tokenizzare con le espressioni regolari 1***

Le espressioni regolari rappresentano uno dei possibili modi di specificare a un algoritmo le regole di tokenizzazione.

Per questo scopo esse si possono usare con diverse strategie.

Nel caso più generale, si compila una serie di espressioni regolari da usare, in un certo ordine, per riconoscere i token nel testo uno per uno (token del tipo parola, numeri, date, ecc.).

Questa strategia consente di raffinare l'algoritmo di tokenizzazione aggiungendo espressioni regolari per includere casi sempre più vari e complessi.

### ***1.11 Tokenizzare con le espressioni regolari 2***

Un approccio più semplice ma decisamente limitato affronta il problema dal verso opposto: specificare espressioni regolari per riconoscere i caratteri o le sequenze di caratteri che servono a separare i token tra di loro.

Per esempio, questa espressione regolare, che fa uso di un meta-carattere "esse" specifica di spezzare il testo in corrispondenza di uno o più caratteri consecutivi del tipo "spazio bianco" (spazio, tabulazione o a capo).

Se proviamo a tokenizzare la nostra frase di test "M'illumino d'immenso") con il tokenizzatore online "WhitespaceTokenizer"... ecco il risultato:

... che come vedi è identico a quello che avevamo ottenuto con TreebankWordTokenizer!

A tale risultato potremmo applicare a cascata quest'altra espressione regolare:

... usandola per isolare come token uno o più caratteri consecutivi del tipo segno di interpunzione o apice o virgolette.

Il risultato finale dovrebbe essere quello del tokenizzatore "WordPunctTokenizer"!

Questo approccio ha il vantaggio di applicarsi in modo simile a molte delle lingue occidentali, compreso l'italiano e il francese, ma funziona solo per un testo "semplice", cioè senza numeri decimali, date, ore, abbreviazioni e senza tutte le ambiguità della punteggiatura di cui ti parlavo poco fa!

### ***1.12 Segmentare un testo in frasi***

Un problema simile a quello della tokenizzazione è la segmentazione di un testo in frasi.

Apparentemente, è una procedura quasi banale. Infatti noi non abbiamo nessuna difficoltà a capire immediatamente dove una frase inizia e finisce.

Il nostro alfabeto usa un carattere apposito: il punto fermo.

Ma la segmentazione in frasi era una necessità prima ancora dell'uso generalizzato del punto.

Torniamo alla nostra epigrafe del sesto secolo: puoi notare un simbolo a forma di foglia di edera (chiamato *hedera distinguensis*) che qui viene usato per separare le frasi senza dover interrompere la scrittura andando a capo.

Perché il marmo era un supporto costoso di cui non andava sprecato neanche un centimetro quadro!

### ***1.13 Il punto nella segmentazione in frasi***

La hedera distinguensis aveva la stessa funzione dei tag “p” e “br” usati nell’Html, il linguaggio del web...

Certo, il punto è più semplice da usare. Ma, come sappiamo, è anche più ambiguo, perché si usa nelle abbreviazioni, nei numeri o nelle sigle. Che quindi costituiscono delle eccezioni alla regola.

Ma niente impedisce che una frase termini proprio con un’abbreviazione o una sigla.

Qui il punto finale, come vedi, costituisce un’eccezione dell’eccezione!

### ***1.14 La segmentazione e il trattamento automatico dei testi***

Nel trattamento automatico dei testi, la segmentazione in frasi è il primo passo per un processo chiamato “allineamento a livello di frase”.

Allineare in questo modo due testi consiste nell’individuare le porzioni di testo corrispondenti. Per esempio, si possono confrontare due edizioni diverse dello stesso testo per verificarne le differenze.

Ma l’applicazione più interessante è nel campo delle traduzioni. Qui l’allineamento consente di collegare i documenti con le loro traduzioni.

Se l’allineamento è a livello di documento, cioè tra un documento le sue traduzioni, si può creare un corpus bi-lingue (o pluri-lingue), come una gigantesca Stele di Rosetta.

E se l’allineamento è a livello di frase, si può confrontare la singola frase nelle diverse lingue.

C’è da aggiungere che, come nel caso della tokenizzazione, non esiste un unico modo “corretto” di segmentare il testo in unità più piccole.

Dipende da che cosa se ne vuole fare!

### ***1.15 L’allineamento e i corpora paralleli***

A cosa serve l’allineamento di testi in più lingue?

Un primo impiego è la produzione di “corpora” paralleli (bi-lingue o pluri-lingue). I corpora paralleli bi-lingue sono grandi raccolte di dati linguistici costituite da coppie di frasi in lingue diverse che sono il primo passo per effettuare automaticamente diverse elaborazioni.

Per esempio:

- studio, anche statistico, delle lingue e delle differenze tra lingue;
- costruzione di vocabolari;
- estrazione di terminologie multilingue.

C'è da aggiungere che grandi corpora paralleli, allineati non a livello di frase, ma a livello di interi documenti o di sezioni di documento, sono stati già prodotti nel corso di secoli.

Il caso più notevole è la Bibbia ebraico-cristiana. Ma ai fini pratici sono molto utili quelli costituiti da atti e documenti di organizzazioni internazionali come l'Onu e l'Unione Europea.

Per esempio, una grande mole di sperimentazione di strumenti linguistici e di produzione di risorse linguistiche è stata effettuata sul corpus Europarl, che contiene gli atti del Parlamento Europeo in 21 lingue.

### ***1.16 L'allineamento e la traduzione***

L'allineamento a livello di frase, effettuato con programmi appositi, è molto utile per alimentare le cosiddette "memorie di traduzione", che sono particolari basi di dati che contengono brani di testo in più lingue.

Le memorie di traduzione sono molto utili per esempio nella traduzione assistita, in cui il traduttore (umano) può verificare se e come una certa frase è stata già tradotta.

In caso affermativo, può riutilizzare la frase così com'è, come nei manuali tecnici che sono molto ripetitivi, o trarne ispirazione o, ancora, verificare criteri alternativi di traduzione.

Naturalmente, anche i sistemi di traduzione automatica fanno ampio uso delle memorie di traduzione. È anche per questo che la traduzione automatica sta iniziando a dare risultati di qualche utilità.

C'è una precisazione da fare: di solito, quando si creano memorie di traduzione, si preferisce produrre segmenti di testo non troppo lunghi, per facilitarne la ricerca e il riuso.

La segmentazione, in questi casi, si può effettuare anche in corrispondenza di segni diversi dal punto fermo, come il punto e virgola, il punto esclamativo, il punto interrogativo.

E, se il testo viene estratto da un documento formattato, ulteriori suggerimenti per la segmentazione possono venire dai tag, come i tag Html di una pagina web.

Così, per esempio, ogni voce di menu può essere estratta come segmento distinto.

