

II Sessione

"Il gusto del progetto per la qualità e la semplificazione normativa"

DISCUSSIONE CON GLI ATTORI DEI PROGETTI

(Stato, regioni, enti locali, operatori)

Elena Bassoli,

Dottoranda di ricerca in metodi e tecniche della formazione e della valutazione delle leggi - XII ciclo

INTRODUZIONE ALL'APPLICAZIONE DI STRUMENTI DI LINGUISTICA COMPUTAZIONALE IN CAMPO GIURIDICO

1. Introduzione

Il presente contributo è suggerito da una più ampia ricerca in corso di svolgimento sulle applicazioni logico-informatiche all'attività di redazione dei testi normativi e intende semplicemente offrire una panoramica sugli aspetti che il più generale tema presenta. In quest'ottica, è possibile che il discorso, di necessità sintetico, sacrifichi talune premesse ed inquadramenti di cui sarebbe utile dare conto. È tuttavia nostra speranza che non ne riesca sminuita la comprensibilità del più generale discorso, che si lega, d'altro canto, alle problematiche che il Seminario ha inteso affrontare nella seconda sessione nella quale l'uso degli strumenti informatici appare affrontato da diverse angolature.

Data la brevità dell'intervento si darà conto dei soli aspetti inerenti la linguistica, tralasciando i concetti ad essa legati, della psicologia cognitiva, a favore, invece, degli aspetti strutturali, quali la sintassi e la semantica, semplificando il discorso.

Con questo lavoro si intende analizzare sommariamente le problematiche legate al linguaggio naturale e al linguaggio giuridico e legislativo in particolare, e la loro formalizzazione ai fini della costruzione di un sistema esperto legale, nonché i linguaggi formali per la rappresentazione della conoscenza. Una particolare attenzione sarà rivolta agli strumenti applicativi inerenti la sintassi e la semantica.

2. Il linguaggio giuridico e i calcolatori

La linguistica computazionale è l'area disciplinare che si fonda su una relazione tra studio teorico della lingua e uso dei calcolatori. Ci sono due modi di correlare linguaggio e calcolatori:

1) il primo consiste nell'utilizzare il calcolatore come strumento potente di memorizzazione, reperimento (1) e manipolazione dell'informazione giuridica. Questa funzione permette di compiere, con più velocità ed efficienza, quelle operazioni che, in ogni modo, il linguista, più specificamente il lessicografo compiva manualmente come la lettura di testi, la schedatura dei passi più rilevanti, l'estrazione delle parole significative, l'identificazione dei contesti e degli esempi utili alla costruzione di un articolo di dizionario (2).

2) l'altro aspetto della linguistica computazionale è fortemente connesso con un uso del calcolatore come modello dello sviluppo del pensiero umano e, nel caso della linguistica, come modello della capacità di comprensione e produzione del linguaggio.

Questa contrapposizione tra le due visioni si collega, tra l'altro, alla nota disputa tra la teoria normativistica teorizzata da Hans Kelsen e il decisionismo sviluppato da Karl Schmitt. Mentre alla prima teoria fa sicuramente riferimento l'informatica giuridica documentaria intesa come capacità di memorizzazione, conservazione e fruizione di informazioni, alla seconda si ricollega l'informatica giuridica decisionale intesa non solo sotto il profilo di creazione delle regole, ma anche come necessità che tale regola sia interpretata per poter poi essere applicata.

Applicare l'uso del calcolatore al trattamento del linguaggio significa soprattutto, saper "pensare come un calcolatore", cioè imparare il metodo di analisi e scomposizione di un problema in termini tali che sia possibile, poi, costruire una sequenza di operazioni semplici (istruzioni) che offrono una soluzione al problema stesso. Per esprimere l'analisi di un problema ad un livello abbastanza formale, orientato alla programmazione, ma non ad uno specifico linguaggio di programmazione, si usano alcuni linguaggi detti

"Linguaggi di specifica" (3).

Una volta compiuta l'analisi formale del problema, si può finalmente passare all'implementazione vera e propria, usando un linguaggio di programmazione scelto tra quelli commercialmente disponibili (4).

I linguaggi di programmazione sono, generalmente, di quattro tipi:

PROCEDURALI quando esplicitano i passi di una procedura nel modo in cui devono essere eseguiti.

FUNZIONALI quando definiscono una funzione che mappa i suoi parametri sul dominio dei risultati. Il linguaggio funzionale per eccellenza è il LISP. La tecnica di richiamare una funzione nel suo stesso corpo si chiama programmazione ricorsiva ed è caratteristica della programmazione in LISP.

DICHIARATIVI quando definiscono il risultato in termini di ciò che è piuttosto che di come si calcola.

A OGGETTI, quando la dichiarazione descrive l'oggetto risultato.

L'interazione uomo-macchina in linguaggio naturale

Dopo il trattamento automatico di testi e la traduzione automatica, l'interazione uomo macchina in linguaggio naturale è uno dei settori più vecchi dell'Intelligenza Artificiale e della Linguistica Computazionale. L'obiettivo generico che stava alla base degli sforzi di ricerca in questo settore era l'assunzione che accedere ai servizi messi a disposizione da un calcolatore (agli inizi soltanto accesso alle basi di dati) nella propria lingua, anziché usando un qualche linguaggio di programmazione, avrebbe reso l'utilizzo del mezzo informatico più facile per un utente impreparato e che, quindi, avrebbe contribuito ad una più rapida accettazione della nuova tecnologia, tra l'altro minimizzando i costi di istruzione per gli addetti. A tal proposito potremmo definire sistemi della **prima generazione** (5), quelli in cui non è integrata alcuna conoscenza linguistica teorica. Essi, infatti, funzionano attraverso il riconoscimento, nelle frasi dell'utente, di schemi fissi o parole chiave, a partire dalle quali costruiscono delle risposte, sempre secondo schemi fissi.

La **seconda generazione** di sistemi di interazione è caratterizzata da due elementi:

a) i sistemi inglobano componenti che riproducono computazionalmente modelli linguistici teoricamente elaborati;

b) adottano tutta la struttura che potremmo definire *a compilatore*. La struttura classica dei *compilatori* di linguaggi di programmazione prevede che ogni stringa sia prima analizzata sintatticamente e successivamente interpretata semanticamente. I due moduli, quindi, il *parser* e l'*interprete semantico* sono rigorosamente sequenziali senza possibilità di integrazione dell'uno nell'altro (6).

Nei sistemi di seconda generazione, se l'interpretazione delle frasi era relativamente ben fatta e soddisfacente sia sotto il punto di vista dell'accuratezza, che della motivazione teorica, l'interazione risultava estremamente fastidiosa, in quanto ciascuna frase veniva interpretata e trattata isolatamente, cosicché, data la domanda

Quanti automobilisti hanno preso una multa nel 1999?

non poteva seguire la domanda

Quanti di essi l'hanno presa per eccesso di velocità?

in quanto *essi* fa riferimento alla domanda precedente, ma può seguire solo

Quanti automobilisti che hanno preso una multa nel 1999 l'hanno presa per eccesso di velocità?

Per ovviare a tale carenza, i sistemi di **terza generazione** introducono alcuni semplici meccanismi di trattamento della coreferenza, sia essa anaforica (pronominale) che realizzata mediante ellissi (7).

Con la **quarta generazione** si ha un vero salto di qualità in quanto si riscontra che il processo di generazione di risposte, almeno negli umani, è fondato non su un calcolo limitato al significato letterale della singola frase, ma è il frutto di una complessa attività di ragionamento, che permette di varcare il limite della letteralità (8).

3. Il linguaggio legislativo

Le conoscenze che un sistema esperto possiede sono espresse in un linguaggio; il passaggio da linguaggio naturale a linguaggio formale incontra numerosi ostacoli e diversi strumenti per superarli. Il linguaggio legislativo ha per la sua stessa funzione, minore ricchezza espressiva di un testo discorsivo. In un testo legislativo si incontrano diversi tipi di enunciati, quelli esprimenti connotazioni applicative (temporali, spaziali, rapporti con normativa preesistente, dichiarazioni di intenti), e quelli più propriamente regolativi (9).

Gli enunciati regolativi sogliono a loro volta essere distinti, secondo la funzione, in enunciati prescrittivi

(norme di condotta) o definitivi. Sia i prescrittivi sia i definitivi presentano una struttura comune, nella quale le conseguenze, cioè il comportamento e il *definiendum*, sono collegate a premesse (condizioni giuridico-fattuali e *definiens*).

Nel processo di formalizzazione tale struttura è espressa nella ormai nota forma logica IF... THEN. Se le condizioni vengono soddisfatte...allora si avrà una determinata conseguenza. È il problema della traduzione in regole, che, una volta risolto, consente di rappresentare la struttura di un testo normativo in un albero decisionale (10).

4. I linguaggi formali di rappresentazione della conoscenza.

Allo scopo di caratterizzare formalmente il linguaggio naturale, occorre collocarlo tra altri linguaggi formalmente definiti, e confrontarlo con essi. Per arrivare all'implementazione del modello nella macchina occorrono ulteriori passaggi che costituiscono il fulcro del procedimento di rappresentazione. Occorre infatti, partendo dal linguaggio naturale in cui si esprime l'esperto, giungere al linguaggio macchina, in un primo tempo eliminando le implicazioni semantiche proprie dei linguaggi naturali, traducendo cioè gli oggetti e le relazioni tra essi in linguaggio formale che rappresenti la struttura sintattica del discorso, e, in un secondo momento, traducendo gli elementi del discorso dal linguaggio artificiale utilizzato in linguaggio macchina, comprensibile al calcolatore, e con cui si possa dialogare. Il linguaggio giuridico naturale, tuttavia, non è facile da trattare poiché spesso il significato delle parole è indefinito e necessita di interpretazioni univoche per le singole parole; inoltre il linguaggio naturale è ambiguo per definizione, mentre il linguaggio formale deve necessariamente essere ben definito sia semanticamente, sia sintatticamente.

Esistono vari tipi di formalismi di rappresentazione, e quello per eccellenza e più adatto al dominio giuridico è la logica. Oltre ad essa i più noti formalismi sono la *Property list*, la rete semantica, i *frames* e le regole di produzione. A questi, che sono riproduzioni interne della realtà nella base di conoscenza, occorre aggiungere la cosiddetta "rappresentazione diretta" o analogica.

Il linguaggio naturale in cui è espressa la legislazione è più facilmente comparabile ad una forma di logica simbolica, piuttosto che ad algoritmi e ciò spiega perché sia preferibile adottare linguaggi di programmazione logica (11), in particolare l'implicazione, anziché linguaggi di programmazione convenzionale.

In tal modo gli enunciati legislativi formalizzati nel linguaggio della logica simbolica sono gli assiomi dai quali il sistema può dedurre conseguenze logiche (12). Peraltro la rappresentazione a regole presenta il limite di fermarsi al primo livello di interpretazione: quello delle relazioni sintattiche fra gli enunciati, a livello quindi di logica proposizionale. La base di conoscenza è costituita da regole che consentono al sistema di operare deduzioni limitate però alla conoscenza fornita, in maniera quindi non intelligente. Per ovviare a tali limiti, ad esempio sarebbe opportuno inserire all'interno della base di conoscenza le informazioni della dottrina giuridica, che è, a differenza del testo legislativo, conoscenza squisitamente descrittiva e va pertanto formalizzata con una struttura diversa, quale la rete semantica (13) che consentirà di sfruttare tale conoscenza nel processo inferenziale.

5. Il Parser e l'analisi sintattica

Al di là del controllo meramente ortografico del testo legislativo, per il quale risulta indispensabile, all'interno del sistema, una quantità indefinita di dizionari, definizionari, thesauri e correttori (14), il componente di maggior rilevanza di ogni sistema di trattamento automatico del linguaggio naturale è l'analizzatore sintattico. L'analizzatore sintattico, o *parser*, è un programma che legge una frase (o un testo una frase alla volta) e, consultando una grammatica opportunamente definita e scritta in un formalismo specifico, restituisce in *output* la struttura sintattica della frase. La tecnica di *parsing* trae origine dai primi tentativi di fare del calcolatore un simulatore del parlante/ascoltatore umano, in particolare dai primi programmi di interazione uomo-macchina in linguaggio naturale (15); al giorno d'oggi è divenuta una tecnica di base per applicazioni di natura sia simulativa che manipolativa (come tecniche avanzate di ricerca di testi e di informazione); i moderni *parsers* restituiscono, a fianco alla struttura sintattica della frase, anche una sua interpretazione semantica, in forma varia.

La scrittura della grammatica è un passo fondamentale nell'utilizzo di un *parser*, anche se è uno *skill*

poco diffuso. Il formalismo di scrittura di una grammatica è evoluto enormemente attraverso i 30 anni di storia del *parsing*, ma la base restano sempre le grammatiche *Context Free*, un tipo di grammatica a struttura sintagmatica, che è stato sviluppato da N. Chomsky (16) come parte della teoria dei linguaggi formali, a sua volta parte della grammatica generativo-trasformativa.

Nei programmi per la comprensione del linguaggio naturale la grammatica viene usata in *parser* (17) separati dagli enunciati di cui si vuole determinare il significato e sui quali avere risposte appropriate. Il *parser* consente di creare una struttura complessa come un albero di derivazione partendo dalla scomposizione di stringhe lineari di parole.

Un *parser* è pertanto un programma che si articola in un numero ristretto di funzioni:

- scansione della frase in *input*: la scansione può avvenire da sinistra a destra o da destra a sinistra. Il metodo più naturale e più usato è da sinistra a destra; il procedimento porta ad esaminare una parola alla volta. In alcuni casi, però, riuscire a esaminare una o due parole più avanti aiuta a ridurre l'ambiguità; in questo caso si dirà che il *parser* è dotato di una *finestra* o di un *look-ahead* di k parole;

- scansione della grammatica e reperimento delle regole rilevanti. Questa fase ha luogo secondo due tecniche principali: partendo dalla regola più alta si procede ad istanziare le regole sempre più basse fino ad incontrare la regola i cui simboli a destra della freccia soddisfano la stringa in *input*; questa tecnica è detta *top-down*.

Oppure, partendo dal simbolo nella frase, si ricerca la o le regole la cui parte destra soddisfi la stringa in *input*; questa tecnica è detta *bottom-up*.

In ambedue i casi, se vi sono scelte alternative, queste possono essere provate una ad una (*depth-first*) o tutte in parallelo (*breadth-first*) (18);

- strutturazione della frase e memorizzazione progressiva (composizionale) di tale struttura. La maggioranza dei *parser* per il linguaggio naturale vengono caratterizzati per questa ultima funzione.

Disponendo di un *parser* l'elemento critico nella costruzione di applicazioni è la scrittura della grammatica, che, in genere, è funzione del tipo di testi da analizzare e, quindi del tipo di applicazione. Una grammatica può essere scritta a partire dalle conoscenze correnti di sintassi, ad esempio prendendo l'insieme delle regole dalla bibliografia corrente, ma, sul terreno applicativo, il modo più efficace consiste nell'estrarre l'insieme delle regole da un *corpus* di testi direttamente connessi con il dominio da trattare.

Il problema più complesso nella scrittura di una grammatica computazionale è la crescita della sua lunghezza e della sua complessità. Man mano che una grammatica viene accresciuta, le interferenze tra regole (*side-effects*) aumentano proporzionalmente, costringendo a complesse procedure di revisione e di mantenimento della consistenza (19).

6. La semantica e la forma logica della frase

Il primo obiettivo che l'interpretazione semantica mira a soddisfare è il reperimento dell'informazione testuale o documentaria; a questo scopo occorre assegnare ad ogni parola isolata una rappresentazione del suo significato e istituire relazioni tra le parole stesse (sinonimia, iperonimia, meronimia, ecc.). Le relazioni lessicali tra le singole parole sono, tradizionalmente, espresse secondo i formalismi proposti dagli psicologi cognitivi, come le reti semantiche *i frames* o le semplici relazioni lessicali.

Per fornire l'interpretazione semantica di una frase, invece, occorre disporre di una descrizione del significato delle singole parole tale che si possa combinare con le parole circostanti in un significato composito, secondo la gerarchia dei sintagmi.

La logica è, come abbiamo visto, la disciplina che offre la base più credibile alla rappresentazione del significato di una frase, soprattutto considerando che fornisce lo strumento automatico per ragionare sul contenuto delle frasi prima di avviarle ad una qualche forma di esecuzione. Inoltre, la logica costituisce uno strumento adeguato anche per la rappresentazione delle relazioni tra parole.

La logica in quanto tale ha per obiettivo la traduzione delle frasi del linguaggio in formule che possono essere trattate da un sistema di ragionamento e per le quali è possibile calcolare una semantica, in termini di un valore di verità (vero/falso). Peraltro nel campo della linguistica computazionale (20), la verità o falsità della formula non è rilevante; acquisisce maggiore rilievo la rappresentazione in termini logici solo per lo schema formale.

Una frase introduce un certo numero di oggetti, tra i quali stabilisce una relazione o dei quali predica

qualcosa. In termini di rappresentazione, il verbo funge da predicato e i suoi complementi sono gli argomenti. Dunque, un prerequisito essenziale per la costruzione della struttura predicativa di una frase è la definizione della cosiddetta *reggenza stretta* del verbo, cioè quanti e quali elementi sono suoi argomenti obbligatori o inerenti (agente, oggetto, ricevente ecc.). La frase

Tizio ruba a Caia un coniglio

può essere tradotta come

rub(a) (Tizio, coniglio... Caia)

che corrisponde alla struttura astratta

$rub(a) (x, y, z)$

I tipi di oggetti che entrano a far parte di questa rappresentazione sono due *costanti*, cioè elementi unici e non generici, (Tizio e Caia) ed un elemento generico, cioè un'istanza dell'insieme dei "conigli". In termini di relazioni lessicali, questo significa che le *costanti* possono essere connesse solo con se stesse, mentre la *variabile* riferita a "coniglio" è in relazione di dipendenza gerarchica rispetto al concetto generico "coniglio". Questa relazione, espressa anche in termini di strutture di conoscenza equivale, a sua volta, ad una predicazione del tipo

$coniglio(y)$

che si interpreta "quell'oggetto x di cui si può predicare che è un coniglio". Un modo alternativo di rappresentazione è

$y:coniglio$

che si interpreta "quell'oggetto y di tipo coniglio". La rappresentazione della frase può divenire, così

$rub(a) (Tizio, y, Caia) \wedge coniglio(y)$

Naturalmente il nostro esempio non esaurisce tutte le possibilità di espressione. Se, ad esempio, si volesse rappresentare la frase:

Tutti i vicini rubarono un coniglio a Caia

secondo i criteri introdotti precedentemente, avremmo

$rub(a) (x, y, Caia) \wedge vicino(x) \wedge coniglio(y)$

dove, però, si perde l'informazione che *i vicini* è plurale. Assegnando a *vicino* la variabile x, la corretta interpretazione è che tutti i valori di x rispondenti alla caratteristica di essere vicini di Caia soddisfano al predicato "rubare un coniglio a Caia". Ci troviamo di fronte ad un'espressione che enumera le istanze di una variabile, cioè le *quantifica*. Il linguaggio naturale conosce molte sfumature (un po', pochi, qualche, ecc.), mentre la logica conosce solamente "tutti" (" per ogni) e "uno" \exists (esiste almeno uno).

Alla luce di questo, la frase precedente diventa:

$\exists x: vicino(x) \wedge rub(a) (x, y, Caia) \wedge coniglio(y)$

da intendersi "per tutte le istanze x essere vicino implica che ruba a Caia un oggetto di cui posso predicare che è un coniglio".

Una frase che, come quella introdotta sopra, sia riducibile ad una variabile quantificata associata ad una predicazione, si dice che è nella "portata" del quantificatore. Se una frase contiene più di un quantificatore, si producono dei fenomeni di ambiguità di "portata". Ad esempio, la frase

Ogni uomo ama una donna

può contenere le seguenti due rappresentazioni

$\forall x \mid uomo(x) \exists y \mid donna(y); ama(x,y)$

che formalizza la lettura per cui "ad ogni uomo corrisponde una donna che lui ama", oppure

$\exists y \mid donna(y) \forall x \mid uomo(x); ama(x, y)$

Con questo si esaurisce la struttura principale della frase. Il linguaggio naturale, però, dispone di una maggior ricchezza di espressione, che pone dei problemi di rappresentazione logica, talora non facilmente risolvibili. Se, ad esempio, si ha la frase

Tizio ruba a Caia un coniglio nero

dovremo completare la rappresentazione data sopra con un riferimento al colore del coniglio, ciò che può essere ottenuto in due modi, o considerando "nero" un attributo qualunque di coniglio, per cui avrò

... $coniglio(y) \wedge nero(y)$

oppure considerando "nero" come il valore di proprietà definita "colore" dell'oggetto coniglio, come segue

... $coniglio(y) \wedge colore(y, nero)$

I sintagmi preposizionali costituiscono, anch'essi, un problema, quando non sono retti direttamente dal

verbo. Una soluzione piuttosto frequente consiste nel considerare la preposizione come un predicato i cui argomenti sono il sintagma nominale dominante e quello che costituisce il PP stesso, come segue
il coniglio di Caia --> *coniglio (y) ^ di (y, Caia)*

Se, poi, la frase includesse un riferimento temporale, come in

Ieri Tizio rubò un coniglio a Caia

la rappresentazione sarebbe complicata dal fatto che l'indice temporale debba essere indicato in qualche modo (21). La soluzione più spesso praticata è quella di considerare anche il verbo, cioè il predicato, come un'istanza di una qualche azione o evento, cui riferire il tempo o qualunque altra determinazione riferita nella frase ad esso, ottenendo, così

ruba (Tizio, y, Caia, e) ^ coniglio (y) ^ ieri (e)

Altri tipi di elementi sarebbero trattabili solo con la logica del secondo ordine.