



UNIVERSITÀ DEGLI STUDI DI MACERATA

DIPARTIMENTO DI STUDI UMANISTICI

CORSO DI DOTTORATO DI RICERCA IN
STORIA LINGUISTICA DELL'EURASIA

CICLO XXVI

TITOLO DELLA TESI

Chatbot e Interfacce Dialoganti: la nuova frontiera

RELATORE

Chiar.mo Prof. DIEGO POLI

DOTTORANDO

Dott. ANDREA SGARIGLIA

COORDINATORE

Chiar.mo Prof. FABIO MASSIMO ZANZOTTO

ANNO 2014

HEMMER

Il sesto membro dell'equipaggio del Discovery non ha dovuto essere messo in condizione di ibernazione perché, in realtà, è l'ultimo ritrovato in fatto di macchine pensanti: il calcolatore H-A-L-9000, che può riprodurre (alcuni esperti preferiscono la parola *imitare*), la maggior parte delle attività del cervello umano con una velocità ed una sicurezza incalcolabilmente maggiori. Abbiamo parlato con il calcolatore H-A-L-9000, al quale ci si rivolge chiamandolo Hal. Buonasera, Hal. Come vanno le cose?

HAL

Buonasera signor Hemmer, tutto va estremamente bene.

HEMMER

Hal, tu hai un'enorme responsabilità in questa missione, in molti sensi forse la maggiore responsabilità di ogni altro membro dell'equipaggio: tu sei il cervello ed il sistema nervoso centrale dell'astronave e le tue responsabilità comprendono la sorveglianza degli uomini ibernati, questo ti causa mai una certa apprensione?

HAL

Possiamo dire questo, signor Hemmer: la serie 9000 è l'elaboratore più sicuro che sia mai stato creato, nessun calcolatore Novemila ha mai commesso un errore o alterato un'informazione. Noi siamo, senza possibili eccezioni di sorta, a prova di errore e incapaci di sbagliare.

HEMMER

Hal, malgrado il tuo immenso talento ti senti mai frustrato dal fatto di dover dipendere da altri per svolgere le tue funzioni?

HAL

Nemmeno minimamente. A me piace lavorare con la gente. Ho rapporti diretti e interessanti con il dottor Poole e il dottor Bowman; le mie

responsabilità coprono tutte le operazioni dell'astronave, quindi sono perennemente occupato, utilizzo le mie capacità nel modo più completo, il che, io credo, è il massimo che qualsiasi entità cosciente possa mai sperare di fare.

HEMMER

Dottor Poole, che cosa pensa del dover vivere per quasi un anno intero in così stretto rapporto con Hal?

FRANK

Ecco... è pressappoco come lei ha detto poco fa: lui è come il sesto membro dell'equipaggio, ci si abitua rapidamente all'idea che parli e lo si considera... beh... come un'altra persona.

HEMMER

Parlando con l'elaboratore si ha l'impressione che esso sia capace di... reazioni emotive. Per esempio: quando l'ho interrogato sulle sue capacità ho sentito un certo orgoglio nella sua risposta circa la sua precisione e perfezione. Lei crede che Hal abbia una genuina emotività?

DAVID

Beh, si comporta come se l'avesse. È stato programmato così, per renderci più facile parlare con lui ma... se abbia davvero una sua propria emotività è una cosa che nessuno può dire con esattezza.¹



¹ Dal film, *2001: Odissea nello spazio* (1968), regia di Stanley Kubrick. La scena è presente all'indirizzo <http://www.youtube.com/watch?v=Y5kkfKLSMkg> di youtube.

Fig. 1 - Hal 9000, il computer di bordo di *2001: Odissea nello spazio*

Introduzione: paradigmi di interfacce dialoganti

Hal 9000, il computer di bordo dell'astronave di *2001: odissea nello spazio*, è probabilmente uno dei più celebri “personaggi” della cinematografia moderna e in un certo senso è entrato a far parte del nostro immaginario collettivo, come uno degli esempi più pregnanti delle possibilità offerte dal mondo informatico. nella prospettiva di una interazione con l'essere umano.

Hal 9000 è un modello di intelligenza artificiale capace di tali abilità nell'elaborazione del linguaggio umano, nella fattispecie della lingua inglese, che risulta in grado di comprendere, parlare e addirittura leggere le labbra.

È oggi estremamente chiaro che Arthur C. Clark, l'autore del racconto di fantascienza, *The Sentinel*,² da cui venne tratto il film di Kubrick risultò estremamente ottimista nella tempistica per il raggiungimento di simili capacità, ma ben lungimirante per quanto riguarda le caratteristiche di un simile prototipo di intelligenza artificiale.

Come Hal 9000 un *agente*³ in grado di interagire con esseri umani attraverso il linguaggio naturale⁴ deve infatti possedere delle capacità di riconoscimento della voce umana (disporre di un apparato uditivo), e al tempo stesso riuscire nella comprensione di quanto acquisito; oltre naturalmente alle interazioni di carattere meccanico e robotico, deve disporre anche di capacità per comunicare con l'ambiente circostante sia in forma scritta, grafica, o anche attraverso dispositivi di sintesi vocale (possedere un apparato fonatorio).

² Clarke Arthur C., *The Sentinel: Masterworks of Science Fiction and Fantasy*. Berkley Books (1983). La prima pubblicazione del racconto è del 1948.

³ Nell'ambito del nostro discorso, con *agente* intendiamo qualsiasi entità in grado di percepire il suo ambiente circostante attraverso dei *sensori* e che al tempo stesso, può interagire con tale ambiente attraverso degli “*effettori*”. Un agente umano ha occhi, orecchie e altri organi che operano prevalentemente come sensori, mentre mani, gambe, la bocca e altre parti del corpo risultano soprattutto effettori. Un agente robotico utilizza ad esempio, telecamere e dispositivi a raggi infrarossi come sensori e vari altri dispositivi meccanici ed elettronici come effettori. Può interagire ovviamente con un essere umano attraverso questi dispositivi.



⁴ Tutti sappiamo di possedere e usare un “linguaggio” che chiameremo lingua naturale. Con tale riferimento intendiamo la capacità che gli esseri umani hanno di sviluppare un sistema di comunicazione con caratteristiche proprie e distinte da altri sistemi di comunicazione. Con lingua intendiamo la forma specifica che questo sistema assume nelle varie società.



Fig. 2 - Il robot "Numero 5" del film *Corto circuito*, di John Badham.

Per tale agente dovranno essere basilari, quindi, ulteriori abilità nella memorizzazione delle informazioni, e nel recupero di queste ultime per elaborarle.

In definitiva, lo studio e il progetto di un simile agente presuppongono competenze estese a ogni livello di studio del linguaggio naturale umano: a partire dalla fonetica, attraverso la morfologia, la sintassi e la semantica, sino alla pragmatica e alla gestione del discorso.

Per comprendere l'obiettivo finale ipotizzabile nella realizzazione di tali dispositivi possiamo far nuovamente riferimento all'esempio di HAL e al tipo di abilità specifiche che un analogo sistema dovrebbe possedere per svolgere il proprio compito.

Innanzitutto, il sistema deve essere in grado di analizzare il segnale audio proveniente dalle corde vocali dell'eventuale interlocutore, in questo caso Dave, e decodificare l'esatta sequenza di parole usate da Dave per produrre quel segnale. Non necessariamente tale segnale può essere acquisito soltanto attraverso l'apparato acustico.

DAVE

Apri la saracinesca esterna, HAL.

HAL

Mi dispiace, purtroppo non posso farlo.

DAVE

Qual è il motivo?

HAL

Credo che tu lo sappia altrettanto bene quanto me. [...]

Dave anche se nella capsula avete preso ogni precauzione perché io non vi udissi, ho letto il movimento delle vostre labbra...⁵

⁵ Dato che HAL è programmato per rimanere efficiente, non può permettere che gli umani lo spengano, ma è programmato anche per non fare del male agli umani (in corrispondenza alle leggi della robotica proposte da Asimov). Entra quindi in conflitto con se stesso, e decide che evitare la sua disattivazione sia più importante che tenere in vita gli astronauti.



Fig. 3 - Immagine tratta dal film *2001: Odissea nello spazio*. Anche in assenza di segnale udibile, il computer HAL riesce a percepire quanto detto dagli astronauti.

In ogni caso, dopo la fase di ricezione, nel produrre la propria risposta, HAL dovrà essere in grado di scegliere una sequenza di parole e generare un segnale acustico di risposta che Dave potrà riconoscere. Entrambe tali attività richiedono competenze nell'ambito della fonetica e della fonologia, e presuppongono la costruzione di un modello con il quale le parole, nella lingua di riferimento dei parlanti, e nell'ambito di una conversazione colloquiale, vengono pronunciate: nel caso specifico di HAL, questi sarà in grado di produrre, ad esempio, le contrazioni *I'm* e *can't* (nella versione originale dice: "I'm sorry Dave, I'm afraid I can't do that") oltre alle forme normali.

Nel produrre simili varianti, nel definire i plurali delle parole e nella coniugazione degli elementi verbali, il sistema deve perciò aver bisogno di conoscenze morfologiche, allo scopo di gestire correttamente la forma e il comportamento delle parole nel contesto.

Per comprendere la raffinatezza di tale sistema basterebbe confrontarne l'espressività con quella del computer Jerry controllato dalla "Sfera" nel romanzo di Michael Crichton: Jerry utilizza raramente le contrazioni e il suo linguaggio è molto meno sofisticato.⁶

Nella sceneggiatura tratta dal romanzo troviamo:

NORMAN

Jerry...

THE SPHERE

It is not necessary to stop. I do not wish it.

BARNES

Well, it's necessary to me.

THE SPHERE

No. It is not possible.

⁶ Crichton Michael, *Sphere*, Knopf (1987). Nel 1998 venne realizzata una versione cinematografica del romanzo per la regia di Barry Levinson.

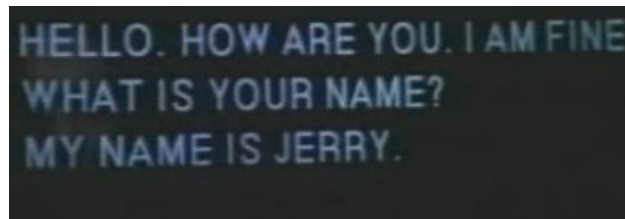


Fig. 4 - Dal film *Sfera*. In questa schermata, Jerry si presenta ai suoi ospiti.

Al di là delle singole parole, Jerry, HAL e gli altri modelli di interfaccia affini devono comunque essere in grado di analizzare la struttura dell'espressione nella quale le varie parole sono inserite: anche solo per comprendere, durante la ricezione, se ciò che viene pronunciato dall'eventuale interlocutore è un ordine oppure una domanda:

HAL, the pod bay door is open.
HAL, the pod bay door is *open*?

Nel caso di una esplicita domanda a cui replicare il computer dovrà costruire a propria volta una stringa di parole connesse tra loro in modo da costituire una risposta. Per realizzare tale scopo sarà necessario utilizzare delle competenze *sintattiche* in grado di permettere la realizzazione di “frasi ben formate” secondo la grammatica con cui la lingua è costruita.

HAL: I'm I do, sorry that afraid Dave I'm can't. [frase non ben formata]
HAL: I'm sorry Dave, I'm afraid I can't do that. [frase ben formata]

Congiuntamente al rispetto delle norme sintattiche, il computer dovrà individuare il significato della propria risposta, significato ricostruibile da quello delle singole parole e dalla maniera con la quale esse verranno a comporsi: dovrà saper ricostruire, partendo dalle strutture che abbiamo visto e dalla rappresentazione del significato di ogni parola, la rappresentazione del significato di ogni frase, combinando i risultati ottenuti in un “tutto” coerente.

Infine avrà il compito di mettere in corrispondenza ciò che è stato effettivamente detto dall'interlocutore, con le intenzioni e gli obiettivi di quest'ultimo.

Nel film *Blade Runner*,⁷ di Ridley Scott, uno dei replicanti protagonisti della storia, Leon, viene interrogato, per scoprire la sua vera identità attraverso un test prevalentemente verbale. Il replicante cerca di comprendere la natura delle domande prima di dare una risposta:

LEON

Inventa lei le domande signor Holden, oppure
gliele scrivono?

⁷ *Blade Runner* (1982), regia di Ridley Scott.

HOLDEN

La testuggine giace sul dorso, la sua pancia arrostitisce al sole rovente, agita le zampe cercando di rigirarsi ma non può, non senza il tuo aiuto, ma tu non l'aiuti.

LEON

Come sarebbe non l'aiuto?

HOLDEN

Sarebbe che non l'aiuti. Perché Leon?

Il replicante rimane in silenzio, in cerca di una risposta plausibile che però non viene: una interfaccia deve avere necessariamente una competenza semantica della lingua utilizzata e al tempo stesso esprimersi veicolando significati al di là della loro correttezza sintattica.

In un altro momento del film, il protagonista Deckard interroga la replicante Rachael:

DECKARD

You are given a calfskin wallet for your birthday...

RACHAEL

I wouldn't accept it, also I'd report the person who gave it to me to the police.

La risposta ha un senso: nella versione originale, sembra però eccessivamente meccanica e difatti il replicante viene individuato. Il linguaggio naturale umano presenta, del resto, come abbiamo accennato, molteplici livelli di complessità, alquanto difficili da gestire congiuntamente.

Ritornando ad HAL, le espressioni nella risposta fornita a Dave: "I'm sorry", "I'm afraid", oltre che rendere più articolata tale risposta, sono testimonianza di un utilizzo specifico della lingua per mostrare il "rispetto" del computer nei confronti dell'essere umano (il ruolo di servizio del computer), e svolgono nella fattispecie una funzione *pragmatica*, cioè relativa a quella disciplina della linguistica che si occupa dell'uso della lingua come azione reale e concreta.

Altro differente esempio in quest'ambito è riscontrabile nell'interfaccia dialogante Joshua, il computer del film *Giocchi di guerra*,⁸ che, in clima di guerra fredda, sollecitato dai dialoghi con il giovane protagonista David, inizia una partita a "Guerra termonucleare globale". Ciò che David crede un videogame è in realtà il sistema di reazione del comando statunitense a un concreto attacco sovietico. David riesce a interrompere l'attacco facendo *praticamente* giocare al sistema, una serie di partite a Tris, il nostro "filetto", un gioco che, come una eventuale guerra nucleare, a partire da determinate configurazioni, non ha vincitori né vinti. Anche Joshua dialoga con David con voce sintetizzata e trascrive il

⁸ *Wargames (Giocchi di guerra)* (1983), regia di John Badham.

contenuto di quest'ultima sullo schermo. Apprende dal Tris una nuova esperienza e conclude riferendosi alla guerra che sta per scatenare: "Strano gioco: l'unica mossa vincente è non giocare."

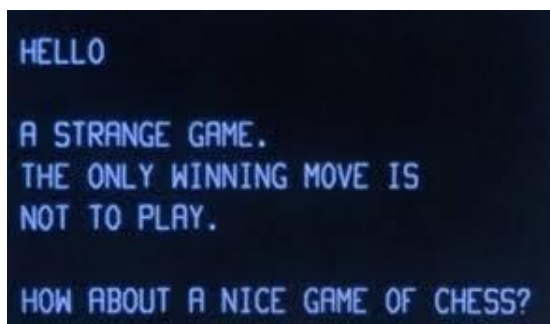


Fig. 5 - Schermata della risposta finale di Joshua, nel film *Wargames*.

Abbiamo visto soltanto alcuni dei vari livelli di competenza linguistica necessari a una interfaccia vocale. Notevolmente importanti sono, per altro, anche le competenze nell'ambito dell'analisi del discorso e dell'analisi testuale (ne parleremo in seguito più in dettaglio), che aggiungono ulteriori gradi di difficoltà per gestire il linguaggio naturale umano nella sua interezza.

Oggi giorno, malgrado le difficoltà di cui abbiamo parlato, molte delle singole tecnologie necessarie sono già disponibili in commercio. I programmi di riconoscimento e di sintesi vocale sono utilizzati e piuttosto diffusi e persino presenti nella maggior parte dei personal computer attualmente prodotti.

Ciò nonostante, quantunque tali interfacce siano sempre più presenti nell'interazione tra l'uomo e il mondo dell'informazione computerizzata, queste ultime non dispongono interamente, al momento, dei requisiti appena descritti e anzi, nella realtà che viviamo, sembrano, forse, "lontane" dal possederli in modo del tutto efficiente.

Lo scopo del mio studio è proprio di analizzare tale realtà, concentrandomi soprattutto nelle tecniche per la realizzazione di sistemi dialoganti artificiali.

A titolo esemplificativo produrrò un esempio pratico di tali sistemi e attraverso il software da me progettato metterò in risalto le problematiche e i principali approcci attualmente escogitati nella realizzazione di simili dispositivi.

Mi concentrerò quindi su due aspetti strettamente connessi al discorso in questione: l'elaborazione di testi da parte di agenti artificiali, l'estrazione di informazioni da tali testi e la produzione, o generazione di testi ulteriori da parte degli stessi agenti: entrambe le discipline, come evidenzierò più in dettaglio, sono alla base, di qualsiasi sistema dialogico tra uomo e *macchina*.

Obiettivi

Prima di proseguire nel discorso è necessaria una breve premessa. Ho difatti impostato questo lavoro di ricerca nel tentativo di perseguire molteplici scopi: innanzi tutto uno scopo essenzialmente divulgativo; definire cioè la programmazione di una interfaccia dialogante anche dal punto di vista di chi non possiede gli strumenti informatici necessari per la sua realizzazione: ad esempio ho ampiamente descritto il contesto storico in cui tale studio si

sviluppa, quali sono i criteri basilari attraverso cui le varie applicazioni vengono realizzate, quali i presupposti strettamente linguistici su cui è necessario far riferimento.

Questo testo, almeno nei miei intenti, può essere letto e compreso, nelle parti essenziali, anche da un non esperto di informatica purché a conoscenza dei requisiti linguistici di riferimento e dei concetti fondamentali messi a disposizione dalla linguistica generale.

Un ulteriore intento del lavoro presentato ha carattere invece strettamente tecnico: ho prodotto due applicazioni di cui la prima in ambiente Windows e la seconda disponibile in rete; allo stesso tempo ho realizzato una serie di librerie o singoli programmi e *strumenti* che possono essere sfruttati come riferimento da chi vorrebbe creare a propria volta una personale interfaccia dialogante.

La terminologia.

In questo lavoro, farò un uso forse “spregiudicato” della lingua inglese e dei termini propri di quest’ultima: difatti il linguaggio informatico è di derivazione essenzialmente inglese e anche le ricerche nel campo della elaborazione del linguaggio naturale sono molto legate a questa lingua. Va parimenti fatto osservare che anche nei casi in cui esistono termini italiani equivalenti a quelli inglesi l’uso iterato di questi ultimi risulta in definitiva oltremodo artefatto, e se ad esempio il cosiddetto *tagging* potrebbe anche essere tradotto come “etichettamento”, farò utilizzo dell’espressione inglese e dei vari costrutti ad essa associati, senza perdermi nei meandri di improbabili, forse cacofoniche, e paradossalmente poco eleganti versioni italiane.⁹

I linguaggi di programmazione utilizzati.

Nella Babele contemporanea dei linguaggi di programmazione ho creduto opportuno gestire una gamma variegata di tali linguaggi effettuando la scelta in base alle librerie disponibili, optando per quelle più esaustive ed esemplificative dei concetti espressi nel mio discorso. Ho privilegiato sviluppare codice in Visual C++ costruito sia sulle librerie della MFC (Microsoft Foundation Class), con il quale ho prevalentemente realizzato il chatbot¹⁰ LUCY, e sia per la piattaforma .NET (Visual C++ CLI/CLR), con il quale ho costruito le varie utilities che presenterò nel mio lavoro.

Il database dei vari chatbot sarà invece prevalentemente composto da file XML.

Ciò nonostante ho utilizzato anche algoritmi scritti in Java, e considerato le notevoli prerogative offerte da Python per l’NLP.

Nell’ambito dell’approccio ai contenuti offerti dalla rete la conoscenza dei linguaggi HTML e ASP.NET o PHP è naturalmente necessaria.

Il chatbot presente in rete è appunto una applicazione in ASP.NET realizzata attraverso il Visual C#.

Come è evidente, a causa della maniera con la quale i vari algoritmi e programmi sono realizzati e studiati, ho creduto opportuno non fossilizzare la ricerca attraverso l’utilizzo di un solo linguaggio: non esiste un linguaggio prevalente e quindi è necessario orientarsi fra diversi codici, adattandoli a problemi specifici, in relazione ovviamente alla vastità delle problematiche offerte dall’elaborazione del linguaggio naturale.

⁹ In una lezione online di Ingegneria del Software ho appena ascoltato la seguente frase nella quale si traduceva in modo forzato l’inglese *slide*, con “trasparente”: il professore universitario diceva “...l’esempio che vi ho riportato, per chiarezza, in questo *trasparente*...”.

¹⁰ Un *chatbot* o *chatbot* è un software progettato per simulare una conversazione intelligente con esseri umani tramite l’uso della voce o di una interfaccia testuale

1. Riferimenti storici

In questo capitolo e nei successivi mi riferirò alla sigla NLP, *Natural Language Processing*, intendendo con essa tutta la gamma di applicazioni, algoritmi e più in generale qualsiasi forma di conoscenza volta all'elaborazione di uno o più aspetti del linguaggio naturale umano.

Come è facile poter riscontrare, senza tali applicazioni buona parte dell'odierno *World Wide Web* e di *Internet* e forse lo stesso mondo dell'informatica in generale, non esisterebbe. Parimenti, ogni genere di attività inerente lo studio dei linguaggi, delle lingue e della loro fruizione, a livello pratico, è influenzato in maniera sempre più preponderante dalle nuove tecnologie e dalle nuove risorse offerte dal progresso informatico.

Si pensi al massiccio utilizzo dei motori di ricerca, si pensi ad esempio alle possibilità offerte da una piattaforma come Google nel "semplice" servizio di traduzione tra le diverse lingue del pianeta.

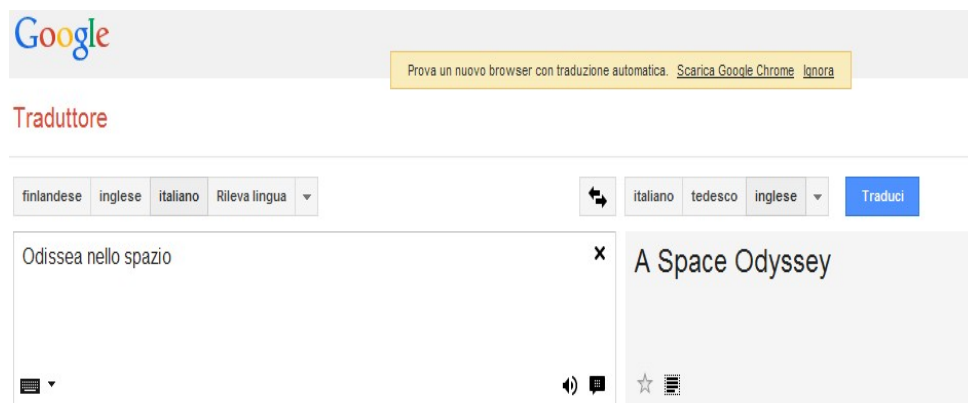


Fig. 6 - Schermata del traduttore online presente sul sito di Google

Il traduttore di Google è un servizio a cui oggi giorno chiunque abbia una connessione internet attiva, un terminale disponibile con un browser funzionante, può ricorrere senza difficoltà.

La possibilità di sfruttare simili risorse è il frutto di un relativamente breve periodo di studi che comprende circa sette decenni, dalla nascita vera e propria dell'informatica come scienza moderna, agli attuali conseguimenti nel campo dell'Intelligenza Artificiale e della Robotica. Quali sono state però le tappe fondamentali di questi progressi?



Fig. 7. - Schermata del sistema di riconoscimento vocale Siri

Il “mito” dell’Intelligenza Artificiale

Talos, nella mitologia greca, era il guardiano di Europa e quindi della sua terra, l’isola di Creta. Forgiato dal divino fabbro Efesto, Talos era un “automa”: una macchina di bronzo che pattugliava la sua terra proteggendola contro nemici e invasori.

Questa sorta di guardiano divino costituisce una delle idee primordiali di vita e intelligenza artificiale.

Sebbene il corpo umano possa essere considerato come una macchina composta da leve, perni e muscoli e quindi, almeno in teoria riproducibile in altre entità meccaniche, agli albori la capacità presunta di creare tali automi era difatti prerogativa esclusiva degli Dei, nessun essere umano era in grado di raggiungere tali conoscenze.

Si consideri ad esempio il mito di Pigmalione raccontato da Ovidio¹¹: lo scultore Pigmalione aveva modellato una statua femminile, nuda e d’avorio della quale si era innamorato considerandola, come tutti gli innamorati, il proprio ideale femminile: una creazione, superiore a qualunque donna, anche in carne e ossa, tanto da dormire accanto a lei sperando che un giorno si animasse.

A questo scopo, nel periodo delle feste rituali in onore di Afrodite, Pigmalione si recò al tempio della dea, pregandola di concedergli per sposa l’essere creato dalle sue mani, rendendola una creatura umana: la dea acconsentì. Lo scultore stesso vide la statua animarsi lentamente, respirare e aprire gli occhi.

Al di là dei miti e delle leggende, uno dei primi automi in senso moderno a essere progettato, fu ad opera di Leonardo Da Vinci: Leonardo realizzò una sorta di androide meccanico, una corazza con all’interno un meccanismo che le permetteva di muoversi come se contenesse un cavaliere. Una entità priva di discernimento e concetto, in grado di realizzare “semplici” spostamenti.

Successivamente si sviluppò la moda degli animali meccanici: sofisticati “oggetti” dedicati soprattutto al divertimento dei vari acquirenti e inventori, piuttosto che a svolgere un lavoro di qualche utilità.

Si pensi alla creazione di Jacques de Vaucanson (XVIII secolo), che realizzò una sorta di musicista in grado di suonare automaticamente il flauto; oppure ci si riferisca alle bambole di Pierre e Henri-Louis Droz, o agli automi sfruttati dai maghi e illusionisti del diciannovesimo secolo per i loro spettacoli. Lo stesso Jean Eugène Robert-Houdini, si serviva di meccanismi nascosti che simulavano movimenti automatici fuori dalle leggi della fisica, ma che in realtà erano prodotti da complici nascosti.

Malgrado questi numerosi esempi, nei quali la componente del linguaggio era pressoché assente, l’idea di una intelligenza applicata a una entità costruita in modo artificiale compare in maniera eclatante con il romanzo di Mary Shelley, *Frankenstein, o il moderno Prometeo*.¹²

Con il Mostro creato dal dottor Frankenstein, il medico-scienziato protagonista del romanzo, l’uomo, seppure attraverso la preesistente materia organica, cioè il corpo e il cervello di persone defunte, è in grado di realizzare ciò che prima era esclusiva prerogativa degli dei: facendo ricorso all’energia presente nei fulmini di una tempesta, riuscirà a forgiare un altro esempio di vita e intelligenza artificiale.

Questa nuova entità, questo *Mostro* (come Frankenstein chiamerà sempre la sua creazione) è in grado, per altro, di apprendere con relativa facilità, di imparare a parlare e

¹¹ Ovidio, *Le metamorfosi*, X, 243-297

¹² Shelley Mary, *Frankenstein, o il moderno Prometeo*, la prima edizione è del 1818, pubblicata in Inghilterra da Lackington, Hughes, Harding, Mavor & Jones.

manifestare nuovi sentimenti, di adattarsi, o meglio cercare di farlo nella realtà in cui ha preso vita. Negli anni in cui Mary Shelley scrive, questa figura costituiva una invenzione “terrificante”, una sorta di storia di fantasmi, ideata proprio con queste intenzioni.¹³

In questo stesso periodo Charles Babbage, inventore e matematico inglese, annunciò alla Royal Astronomical Society che avrebbe presto costruito un prototipo di macchina in grado di svolgere calcoli complessi come i logaritmi: nel 1822 presentò effettivamente alla Society una macchina in grado di risolvere equazioni polinomiali.

Nella convinzione di Babbage, qualora una macchina fosse stata in grado di realizzare un certo tipo di calcoli, questo ipotetico *calcolatore*, poteva, quasi sicuramente, compiere qualsiasi tipo di elaborazione.¹⁴ A proposito della sua “Macchina Analitica” (*Analytical Engine*) scrisse: “le vie attraverso cui sono giunto a questa macchina sonostate le più intricate e complesse mai percorse da mente umana”.¹⁵ Tale macchina doveva essere costruita con migliaia di cilindri dentati ingranati tra di loro in modi incredibilmente complessi. Babbage vedeva un vorticoso entrare e uscire di numeri dai vari moduli del suo progetto, sotto il controllo di un programma contenuto in schede perforate; quest’ultima idea era ispirata dal telaio Jacquard, un telaio controllato da schede capace di tessere disegni di sorprendente complessità. Il progetto di Babbage sebbene rimasto incompiuto, fu finanziato inizialmente e fino alla morte di quest’ultima, dalla contessa Ada Lovelace, ossia Augusta Ada Byron, l’unica figlia del poeta lord Byron, amico di Percy Bysshe Shelley, il padre appunto di Mary Shelley, la creatrice di Frankenstein.

Lady Lovelace, al pari di Babbage, era profondamente consapevole che l’invenzione della macchina Analitica rappresentava per l’umanità un primo passo verso l’intelligenza meccanizzata, specie se la macchina fosse stata capace di “mangiare la propria coda” (in questo modo Babbage descriveva quanto accade quando una macchina interviene sul proprio programma immagazzinato, cambiandolo”), anticipando tematiche in ambito logico, linguistico e matematico che verranno affrontate e per certi versi risolte nel secolo successivo dal matematico Kurt Gödel.

Dal mito alla storia

¹³ Nel maggio 1816 la sorellastra di Mary Shelley, Claire Clairmont, diventata l’amante di lord Byron, convince i coniugi Shelley a seguirla a Ginevra. Il tempo piovoso confina spesso i dimoranti in albergo e questi occupano il tempo libero leggendo storie tedesche di fantasmi, tradotte in francese. Byron propone allora di comporre loro stessi una storia di fantasmi (tra i presenti c’è anche Angelo Polidori, lo scrittore di *Il vampiro*, il precursore di Dracula); tutti cominciano a scrivere, ma Mary non ha subito l’ispirazione. Per altro le lunghe conversazioni degli uomini vertono sulla natura dei principi della vita, su Darwin, il galvanismo, sulla possibilità di assemblare una creatura e infondere in essa la vita. Tali pensieri scatenano l’immaginazione di Mary e le procurano un incubo: sogna un giovane *studente* in ginocchio di fianco alla creatura che ha assemblato; creatura che, grazie a una qualche forza, comincia a mostrare segni di vita.

Mary inizia il racconto decisa a ricreare il terrore che essa stessa ha provato nell’incubo: dopo il successo iniziale, lo scienziato fugge dalla sua creazione, abbandona il *mostro* a se stesso, sperando nella sua morte; ma la creatura, che ha imparato ad esprimersi nel nostro linguaggio, rimane sconcertata dalla propria solitudine (“Satana aveva i suoi compagni che lo ammirassero e incoraggiassero; ma io sono solo”) e cerca una compagna con cui interloquire. A proposito di Frankenstein ricordiamo anche la figura di Karel Čapek che nel 1920 con il dramma in tre atti R. U. R. (*Rossumovi univerzální roboti*) conierà la parola robot (dal ceco robota, “lavoro duro, lavoro forzato”): i *roboti* di Čapek non sono in realtà *robot* nel senso poi attribuito al termine, ovvero automi meccanici, ma esseri “costruiti” producendo artificialmente le diverse parti del corpo e assemblandole insieme.

¹⁴ Babbage fu il primo a ipotizzare la creazione di un calcolatore programmabile per eseguire ogni genere di calcolo. I fondamenti di tale intuizione furono dimostrati matematicamente, quasi un secolo più tardi dal genio di Alan Turing.

¹⁵ Charles Babbage, *Passages from the Life of a Philosopher*, pp. 145-46, Longman (1864).

A distanza di poco più di cento anni, a ridosso della seconda guerra mondiale, la maniera di concepire determinate invenzioni e di immaginare una vita artificiale cambia radicalmente. Viene realizzato uno dei primi esempi funzionali e concretamente utilizzabili di calcolatore elettronico: Colossus, una sorta di primordiale computer tenuto a lungo segreto dalla Gran Bretagna, in grado di decodificare i messaggi delle forze armate tedesche, cifrati con la segretissima macchina *Enigma*. A contribuire nella realizzazione di Colossus è un matematico inglese, Alan Turing¹⁶ colui che è ritenuto il padre della moderna Intelligenza Artificiale e che nel periodo del conflitto si adoperò per mettere le proprie conoscenze al servizio delle forze alleate: Colossus era in grado di svolgere automaticamente calcoli che altrimenti avrebbero richiesto quantità di tempo enormi per un essere umano. L'esempio di Turing è uno dei vari esempi, forse uno dei più significativi, di come la conoscenza matematica e "informatica" abbia un ruolo determinante e imprescindibile per le vicende dell'umanità. La guerra nell'Atlantico contro i sottomarini tedeschi poté essere vinta essenzialmente grazie al contributo degli scienziati che realizzarono Colossus.



Fig. 8 - Howard Aiken il creatore di uno dei primi calcolatori elettronici, Mark 1, presso l'Università di Harvard nel 1944.

Le esigenze belliche fornirono uno stimolo eccezionale allo studio di macchine capaci di gestire una sempre maggiore quantità di calcoli. Nel 1946 si sarebbe arrivati al primo vero e proprio elaboratore elettronico, l'ENIAC.

In questi anni, suggestionanti per la loro rivoluzionaria importanza, si pensò che le prime forme di computer, costituite da parti essenzialmente meccaniche ed elettroniche, potessero indicare la vera strada da percorrere per la realizzazione di una vita e una intelligenza artificiale.

Difatti, al termine della Seconda Guerra mondiale, lo scoppio della cosiddetta Guerra fredda tra i due schieramenti sottoposti agli Stati Uniti e all'Unione Sovietica determinò un ricorso sempre più massiccio (esteso a molte altre discipline scientifiche) al potenziamento dei calcolatori elettronici e con essi, di pari passo, a tecnologie in grado di gestire e elaborare l'enorme flusso di informazioni provenienti dalla parte rispettivamente avversa della cortina di ferro: in Occidente si definì prioritaria la codifica, la decodifica e la classificazione di documenti scritti o trascritti dal russo. L'elaborazione del linguaggio naturale (appunto *Natural Language Processing* o NLP) ad opera di calcolatori elettronici si prospettò come una nuova scienza con grandi possibilità di sviluppo.

¹⁶ Turing Alan Mathison (1912 - 1954), è stato un matematico, logico e crittografo britannico, considerato uno dei padri dell'informatica e uno dei più grandi matematici del XX secolo.

Tale sviluppo si verifica *paradossalmente* dopo che Gödel aveva già dimostrato nel 1931, il suo celebre Teorema di Incompletezza, nello scritto *Sulle proposizioni formalmente indecidibili dei "Principia Mathematica"*¹⁷ e di sistemi affini, confermato per certi versi nella teoria della calcolabilità scoperta da Turing, rivelante l'esistenza di "buchi" inevitabili perfino nel calcolatore più potente che si possa immaginare.¹⁸

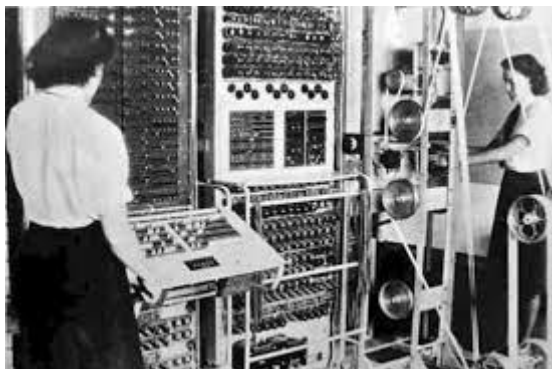


Fig. 9 - Possiamo osservare l'ingombranza e la complessità meccanica ed elettronica dei primi dispositivi.

A prescindere dai fondamenti teorici, e nonostante le molteplici difficoltà in grado di arenare i vari progetti dopo le speranze iniziali, i primi elaboratori furono considerati, in una prospettiva decisamente ottimistica, una valida risorsa per la traduzione automatica dei testi. In questo periodo anche le semplici procedure di archiviazione e lettura dei dati erano complesse ed elaborate: le parole di un vocabolario venivano indicizzate su un nastro e ad esse venivano assegnati dei codici numerici. Agli equivalenti di quelle parole in russo, francese o tedesco, venivano assegnati gli stessi codici.

Per tradurre dal russo all'inglese, un altro nastro coi codici relativi al russo veniva inserito in un computer che ricercava le corrispondenze dei codici e stampava il risultato in inglese.

Tra le figure più importanti, in questo contesto, risulta lo statunitense Warren Weaver,¹⁹ uno dei precursori della cosiddetta traduzione automatica. Weaver aveva constatato come gli elaboratori elettronici, nel corso della Seconda guerra mondiale erano utilizzati essenzialmente, oltre che per calcolare il percorso delle parabole degli ordigni bellici, anche per decifrare i codici del nemico. Considerò quindi il problema della traduzione come una sorta di problema di decifrazione, applicando i metodi già sfruttati per i messaggi in codice alla traduzione dal russo. Dinanzi a un testo russo si poteva effettivamente pensare: questo testo è scritto in inglese ma è stato codificato in una sequenza di strani simboli. Dobbiamo procedere alla decodifica.

¹⁷ Gödel Kurt, "Über Formal Unentscheidbare Sätze der *Principia Mathematica* und Verwandter System, I", *Monatshefte für Mathematik und Physik*, 38, pp. 173-198 (1931).

¹⁸ Tali intuizioni, sull'esistenza di paradossi del linguaggio irrisolvibili e sulla "non calcolabilità dei calcolatori" possono essere riassunte dalle conclusioni di Ludwig Wittgenstein (allievo di Russell, autore dei *Principia Mathematica*) sulla cosiddetta "indicibilità": a tema con i nostri ragionamenti e "stravolgendo" le sue stesse conclusioni secondo cui i limiti del linguaggio sono i limiti del mondo, Wittgenstein, definisce l'indicibile, paradossalmente, proprio solo all'interno della categoria del linguaggio.

¹⁹ Warren Weaver (1894 - 1978) è stato uno scienziato e matematico statunitense. È generalmente considerato uno dei padri della traduzione automatica, ma ha svolto anche un'importante ruolo nella divulgazione della ricerca scientifica negli Stati Uniti.

In realtà tale approccio non venne accettato negli anni '50 e '60 ma, come vedremo, risultò di grande importanza successivamente.

In questi anni, negli stabilimenti dell'IBM vennero, comunque, realizzati esperimenti oltremodo significativi attraverso il mainframe IBM-701. Come vedremo i metodi di traduzione sfruttavano una combinazione di regole sintattiche e grammaticali e di considerazioni statistiche. L'importanza dell'IBM in questa prima era dell'informatica può essere riscontrata anche nella *decodificata* citazione cinematografica del già menzionato computer HAL di *2001: Odissea nello spazio* (il nome HAL non è altro che una derivazione della sigla IBM codificata secondo il "cifrarario di Cesare" spostando le lettere IBM indietro di una posizione: I diventa H, B diventa A, M diventa L: HAL)

Naturalmente, nonostante i problemi iniziali gli scienziati, per la maggior parte, erano concordi riguardo una risoluzione di questi stessi problemi nell'arco di tre o quattro decenni.

Gli anni dal 1964 al 1966 si dimostrarono difatti prolifici nell'ambito degli studi riguardanti l'Intelligenza Artificiale nel suo complesso. Nel 1956 ebbe luogo la Conferenza di Dartmouth nel New Hampshire e proprio qui John McCarthy coniò il termine "Artificial Intelligence". Soltanto un anno dopo, nel 1957, Noam Chomsky pubblicò, *Syntactic Structures*, testo che possiamo considerare tuttora una sorta di rivoluzione nella maniera di concepire il linguaggio naturale umano: l'idea di un nuovo tipo di grammatica per comprendere la struttura delle frasi e la loro formazione. Chomsky parlò di *Phrase-Structure Grammar* e propose il suo modello per una gestione informatica del linguaggio naturale. Nel 1958 John McCarthy progettò il linguaggio di programmazione LISP (Locator/Identifier Separation Protocol) un linguaggio in grado di formalizzare le proprie idee in un contesto pratico. Nel 1964 John Weizenbaum realizzò ELIZA, comunemente considerato il primo chatbot: ELIZA si basa, come vedremo meglio in seguito, su una serie relativamente limitata di regole di ricostruzione delle frasi che per certi versi ricordano le regole di trasformazione teorizzate da Chomsky. Così come le regole di produzione di Chomsky ricordano la definizione delle funzioni *First-Class* effettuabile in Lisp.²⁰

Il proliferare di questo genere di studi sembra per altro in contraddizione con la semplicità delle risorse informatiche a disposizione: negli anni sessanta ci troviamo ancora

²⁰ Un esempio di programma in Lisp:

```
(defun sentence () (append (noun-phrase) (verb-phrase)))
(defun noun-phrase () (append ( Article ) (Noun)))
(defun verb-phrase () (append (Verb) (noun-phrase)))
(defun Article () (one-of '(the a)))
(defun Noun () (one-of '(man ball woman table)))
(defun Verb () (one-of '(hit took saw liked)))
```

Esempio di produzioni in una Grammatica generativo-trasformazionale:

```
F= SN + SV
SN=Art+N
SV=V+SN
Art={the, a}
N={man, ball, woman, table}
V={hit, took, saw, liked}
```

La pressoché totale corrispondenza delle due differenti strutture è una notevole testimonianza della evidente penetrazione delle due discipline.

nell'era delle schede perforate e del batch processing²¹ ma a mio avviso proprio la consapevolezza di trovarsi solamente nella preistoria dello sviluppo dell'informatica stimolava le personalità più brillanti a non porre limiti alle proprie visioni e alla propria immaginazione. Si supponeva soltanto che le difficoltà tecniche sarebbero state un giorno superate e per quel giorno le nuove idee avrebbero potuto finalmente concretizzarsi.

Ciò nonostante la fondazione negli Stati Uniti dell'Automatic Language Processing Advisory Committee, (ALPAC) avvenuta nel 1964, una fondazione con il compito di valutare il grado di progresso degli studi informatici nell'ambito del linguaggio naturale, ben prestò frenò gli entusiasmi iniziali e le ricerche sulla traduzione artificiale furono in un certo senso arrestate poiché dopo i primi lusinghieri successi, dopo dodici anni e milioni di dollari investiti nella ricerca, si giunse alla conclusione che la traduzione effettuata ad opera dell'uomo era di gran lunga *meno costosa* di quella automatica.

Il primo passo indietro nell'ambito dell'IA si verificò quindi per una questione di fondi e per il fallimento dei progetti nel solo campo della traduzione.

Dopo le decisioni del 1966 ad opera dell'ALPAC si verificò un arresto nell'ambito delle ricerche sul *Natural Language Processing*: occorsero più di quindici anni prima che tale campo di studi richiamasse l'interesse delle principali istituzioni e dei finanziatori. La combinazione di ricerche a carattere linguistico e statistico che aveva caratterizzato l'attività degli studiosi negli anni precedenti venne ridimensionata a favore dei soli studi di carattere statistico: importante risultò il contributo di Frederick Jelinek che impostò il lavoro di elaborazione del linguaggio naturale proprio esclusivamente su basi probabilistiche e statistiche. Suo il celebre detto "Every time I fire a linguist, the performance of our speech recognition system goes up."²²

Naturalmente, in questo periodo, significativi miglioramenti tecnologici nell'ambito dell'informatica avevano aperto un nuovo scenario per gli studiosi e per la realizzazione di ulteriori programmi in ambito linguistico.

È il caso di SHRDLU, un progetto terminato nel 1970, in grado di reagire a messaggi linguistici operando²³ in un "mondo" costituito da pochi oggetti colorati in uno spazio circoscritto. SHRDLU evidenziava la possibilità di rivolgere le conclusioni linguistiche all'interno di un contesto limitato ma pur sempre esterno al "mondo" delle semplici "parole" come poteva accadere nell'attività di traduzione. SHRDLU proponeva cioè la possibilità di utilizzare il linguaggio per interagire con la realtà fisica circostante.

Nel 1982 venne concretizzato il concetto di chatbot e iniziò il progetto Jabberwacky. Il progetto si inseriva sempre nella prospettiva di realizzare un programma di Intelligenza Artificiale in grado di simulare il comportamento umano all'interno di una conversazione, gestita in maniera anche divertente e con un approccio umoristico.

La meta prefissata e sognata era ovviamente quella di superare il "famigerato" Test di Turing.²⁴

²¹ Il termine *batch* risale all'epoca della programmazione con schede perforate. In tale contesto, i programmatori solitamente non avevano accesso diretto al computer, bensì preparavano i propri programmi "off-line" e passandoli a un amministratore di sistema, il quale aveva il compito di mandarli in esecuzione appena possibile: solo in seguito i risultati dell'elaborazione erano restituiti agli interessati.

²² Fred Jelinek (1932 - 2010) fu un ricercatore nato in Cecoslovacchia, ma trasferitosi in seguito alla Seconda guerra mondiale negli Stati Uniti: attivo nell'ambito della teoria dell'informazione, del riconoscimento vocale e dell'NLP in genere.

²³ Le varie operazioni erano ovviamente soltanto simulate.

²⁴ La versione presentata da Turing è la seguente: vi sono tre giocatori: un uomo (A), una donna (B) e un interrogante (C) che può essere dell'uno o dell'altro sesso. L'interrogante sta in una stanza da solo, separato dagli altri

A partire dagli anni novanta, gli studi nel campo dell’NLP conobbero una nuova e straordinaria fioritura. La grande quantità di informazioni che fluivano nel World Wide Web stimolò la progettazione di programmi e algoritmi in grado di gestire tale flusso di informazioni, di organizzarle e se necessario riassumerle. Negli Stati Uniti la realizzazione di internet e soprattutto l’enorme quantità di testi che nel vicino Canada venivano prodotti sia in inglese sia in francese, nonostante il termine della guerra fredda, stimolò una nuova ondata di interesse nei confronti della traduzione automatica.

Oltre alle nuove risorse e alla gestione di questa vasta gamma di informazioni si verificò un sensibile avanzamento nell’ambito della sintesi e del riconoscimento vocale. In questo nuovo contesto il governo degli Stati Uniti ricominciò a investire nel settore dell’NLP attraverso tutta una nuova serie di programmi.

Nel periodo attuale, questo tipo di ricerche conoscono un ritrovato fulgore. L’aumento delle potenzialità degli elaboratori e delle capacità di memoria disponibili anche a livello di singoli acquirenti (nel mercato sono disponibili memorie dell’ordine del terabyte: 10^{12} bytes quando fino a pochi anni fa l’ordine di grandezza delle dimensioni degli harddisk si aggirava su pochi gigabyte di memoria - 10^6 bytes).

Grazie a tali risorse e alla disponibilità di una notevole gamma di programmi di gestione dei testi e delle informazioni si sono moltiplicati anche i vari tipi di strumenti messi a disposizione dell’utente: dai semplici programmi di videoscrittura (con già inseriti all’interno strumenti per la ricerca sinonimica, la correzione ortografica e grammaticale), a quelli già citati di sintesi vocale per la video lettura, ai recenti programmi di riconoscimento e dettatura, come la gamma di applicazioni della Dragon Dictate o il programma SIRI messo a disposizione dalla Apple nei suoi smartphone. Altre case produttrici di computer, come la IBM si sono a loro volta cimentate nell’ambito di sfide riguardanti maggiormente l’Intelligenza Artificiale e, ad esempio, con il progetto Watson gli scienziati della IBM sono riusciti a creare una sorta di interfaccia dialogante in grado di competere nel celebre programma di quiz televisivo *Jeopardy*.

Le moderne conquiste nell’ambito dell’NLP consistono quindi principalmente in programmi di riconoscimento vocale, machine learning,²⁵ sintesi vocale, e traduzione automatica. Nel prossimo futuro con la rivoluzione offerta dal Web 2.0²⁶ i computer saranno in grado di interpretare domande e ricercare nella rete le informazioni più assimilabili alle domande, poste in una versione più ampia della ricerca per parole chiave, attualmente disponibile nei principali motori di ricerca.

due. Scopo del gioco per l’interrogante è quello di determinare quale delle altre due persone sia l’uomo e quale la donna. Per evitare che il tono della voce possa aiutare l’interrogante, le risposte dovrebbero essere scritte, o meglio ancora battute a macchina. La soluzione migliore sarebbe quella di collegare le due stanze con una telescrivente. Chiediamoci ora: “Che cosa accadrà se in questo gioco, una macchina prenderà il posto di A?”.

²⁵ L’*apprendimento automatico* (noto come *Machine Learning*) rappresenta una delle aree fondamentali dell’intelligenza artificiale e si occupa della realizzazione di sistemi e algoritmi costruiti in base a osservazioni preesistenti, usate come dati per la sintesi di nuova conoscenza. L’apprendimento può avvenire “catturando” caratteristiche di interesse provenienti da esempi, strutture dati o sensori, analizzando e valutando le relazioni tra le variabili osservate.

²⁶ Il *Web 2.0* è un termine utilizzato per indicare uno stato dell’evoluzione del World Wide Web, rispetto a una condizione precedente. Si indica come Web 2.0 l’insieme di tutte quelle applicazioni online che permettono un elevato livello di interazione tra il sito web e l’utente: i blog, i forum, le chat, i wiki, le piattaforme di condivisione di media come Flickr, YouTube, Vimeo, i social network come Facebook, Myspace, Twitter, Google+, LinkedIn, Foursquare, ecc.; tali piattaforme sono ottenute, tipicamente, attraverso opportune tecniche di programmazione Web e relative applicazioni legate al paradigma del Web dinamico (le applicazioni Web che interagiscono attivamente con l’utente modificando le informazioni mostrate in base alle informazioni ricevute dall’utente stesso) in contrapposizione al cosiddetto Web statico o Web 1.0.

Ovviamente stiamo vivendo questa rivoluzione proprio in questi anni e non è da escludere che le prospettive fin troppo ottimistiche negli anni '50 possano essere attuabili nei prossimi decenni nelle forme prospettate da tanti scrittori e autori di fantascienza.

Va per altro ribadito che proprio in virtù di questi nuovi sistemi una nuova “era” è praticamente già iniziata, una nuova epoca in cui uno dei tratti distintivi tra ciò che è umano e ciò che non lo è, viene progressivamente abbattuto. Nelle parole di Kenneth Colby: “before there were computers, we could distinguish persons from non-persons on the basis of an ability to participate in conversations. But now, we have hybrids operating between persons and non persons with whom we can talk in ordinary language.”²⁷

²⁷ Colby K., *Comments on human-computer conversation*, in Wilks, Y., *Machine Conversations*. pp. 5-8 (1999).

2. Il riconoscimento vocale

Caratteristiche generali.

Sebbene il mio *lavoro* si sviluppi dal presupposto di avere a disposizione un programma in grado di realizzare il riconoscimento vocale (*Speech Recognition: SR*) di quanto espresso da un eventuale utente, è opportuno effettuare una breve premessa sulle caratteristiche del software considerato, su come i vari programmi di questo tipo si sono evoluti e sulle loro applicazioni, anche indipendentemente dalla realizzazione di una interfaccia vocalica.

Il riconoscimento vocale è, innanzi tutto, il processo con cui il linguaggio orale umano viene riconosciuto e successivamente elaborato attraverso un computer o più specificamente attraverso un apposito sistema di elaborazione che traduce i foni pronunciati dall'utente in segni grafici correlati al linguaggio utilizzato: per una lingua come l'italiano, i foni vengono tradotti in morfemi corrispondenti alle parole del vocabolario italiano e al relativo alfabeto.

Le difficoltà incontrate nel costruire questi programmi sono legate sostanzialmente alla relativa lentezza nella formazione di zone di invarianza²⁸ per i suoni linguistici. Difatti quello che noi conosciamo normalmente come lo stesso suono, se osservato dal punto di vista acustico, ci appare nella maggioranza dei casi tutt'altro che identico, tanto da risultare spesso simile ad altri suoni, oppure anche del tutto assente in un'altra configurazione sonora di riferimento. Tale variabilità è anche dipendente dall'interlocutore, che può non avere una corretta pronuncia a causa della sua età, per problemi all'apparato fono-articolatorio, o per via di non prevedibili inflessioni dialettali.

²⁸ Il concetto di invarianza è piuttosto complesso ma basilare per quanto riguarda i processi intellettivi. Facciamo riferimento al viso di una persona che conosciamo: possiamo in realtà individuare quel viso ogni volta che è percepito dai nostri occhi. Accade automaticamente in meno di un secondo, non importa se è a un metro o due metri da noi. Quando la persona è vicina l'immagine del viso occupa gran parte della retina, quando è lontana ne occupa una porzione minore. Può starci di fronte, di profilo, quel viso può ridere, o sbadigliare: alla luce o all'ombra. Quel viso può apparire in un numero infinito di posizioni e varianti.

La rappresentazione che il cervello ha al proprio interno di quel viso è chiamata *rappresentazione invariante*. A livello della neocorteccia cerebrale, monitorando l'attività delle cellule nell'area di cervello utilizzata per il riconoscimento visivo, esiste un settore che rimane costantemente attivo fintanto che quel particolare viso è nelle vicinanze: questa stabilità nelle cellule attivate è appunto la rappresentazione invariante. Questo chiarimento relativo al campo visivo può essere riferito a ogni altro nostro senso, compreso ovviamente l'udito. Possiamo individuare una parola indipendentemente dall'altezza e dalla frequenza con cui questa è stata pronunciata dall'apparato fono-articolatorio. L'apprendimento umano, in sostanza, si verifica mediante la formazione di queste zone di invarianza e nella loro progressiva modifica. Analogò è il cosiddetto "principio del prototipo": l'evento più particolare può servire come esempio generale di una classe di eventi. In poche parole certi eventi particolari si imprimono così fortemente nella memoria che in seguito essi possono essere usati come modelli di altri eventi in qualche modo simili. Dunque, in ogni evento specifico c'è il germe di un'intera classe di eventi analoghi.

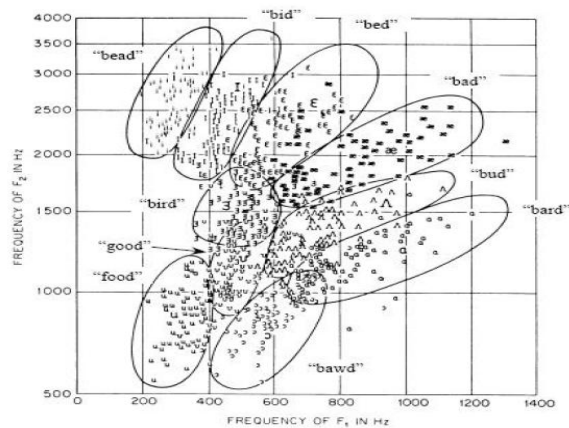


Fig. 10 - Diagramma della variabilità di pronuncia in alcune parole inglesi

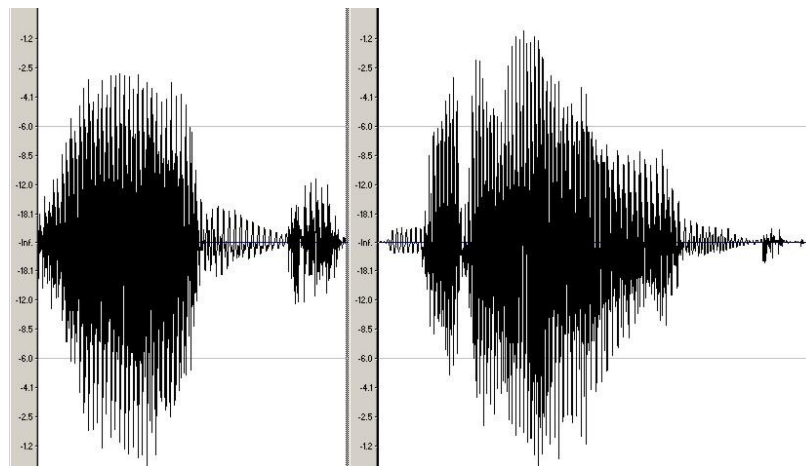


Fig. 11 - Relativa similarità nelle forme d'onda delle parole "egg" a sinistra e "bread" a destra.

Per ovviare a questi sostanziali ostacoli, oltre ai progressivi miglioramenti degli apparati fisici di ricezione e di immediata elaborazione si è fatto ricorso ad algoritmi a carattere statistico in grado di fornire un modello di riferimento dei termini e delle espressioni di una determinata lingua allo scopo di restringere il campo delle scelte nell'individuare i vari foni. Ad esempio nella frase "All omelettes have ___ that are blended" il sistema potrà individuare con molta probabilità in "eggs" la parola corrispondente alla forma d'onda escludendo "bread" dalla scelta.

Al di là di questa fase eminentemente "ricettiva", è importante, per altro, distinguere il problema del riconoscimento vocale da quello della comprensione del linguaggio. Un sistema di riconoscimento vocale non ha bisogno di possedere alcuna forma di comprensione del linguaggio. Per esempio, computer con i controlli vocali attualmente presenti nel mercato non possiedono alcuna forma di comprensione del linguaggio naturale. Piuttosto, le parole riconosciute sono usate come "ordini": *codici* simili ai segnali che vengono mandati con il telecomando verso un televisore. Il riconoscimento vocale si interessa soltanto dell'identificazione delle parole espresse da una determinata sorgente di segnale, senza badare a come tali parole vengono utilizzate per comunicare. Per essere un sistema di comprensione del linguaggio naturale, il riconoscitore vocale dovrà comunicare il proprio input a un diverso sistema in grado di realizzare tale attività.

Dispositivi e programmi di riconoscimento vocale vengono oggigiorno utilizzati comunemente per applicazioni automatizzate nel contesto delle applicazioni telefoniche (ad

esempio nei call center automatici), per sistemi di dettatura che consentono di dettare testi al computer, oppure per altri sistemi di controllo (come nei navigatori satellitari).

Riferimenti storici.

Il primo sistema di riconoscimento vocale vede la luce nel 1952: consisteva in un dispositivo per il riconoscimento di singole cifre parlate; un altro dispositivo, l'IBM Shoebox, venne esposto al Salone di New York del 1964.

I vari programmi di riconoscimento vennero sviluppati inizialmente allo scopo di soppiantare i tradizionali metodi di trascrizione, ma a causa di inadeguate prestazioni, stentaronο ad essere impiegati in modo produttivo.

Soltanto sostanziali perfezionamenti tecnici e la progressiva riduzione degli errori di riconoscimento e dei tempi di training vocalico²⁹ ha garantito il successivo, diffuso utilizzo di questi sistemi .

Caratteristiche specifiche

I programmi in questione funzionano con algoritmi adattativi di tipo statistico inferenziale, che ricostruiscono il linguaggio in base alla frequenza delle associazioni fra parole.³⁰ Gestiscono un vocabolario fonetico con la pronuncia di numerose parole nella lingua selezionata e riconoscono anche il timbro di voce del parlante.

Permettono di dettare documenti in qualunque editor di testo come Notepad, Word, Writer di OpenOffice. Con tali applicazioni è possibile costruire macro³¹ vocali che comandano la freccia del mouse per riavviare il computer, aprire e chiudere file e programmi, accedere ai vari menu o a un dato sito Internet.

Una installazione efficiente richiede in genere la lettura di alcuni brani di prova davanti al microfono del sistema, per abituare il programma a riconoscere la voce di chi lo utilizzerà: le registrazioni vengono memorizzate e analizzate per costruire una libreria di file vocali (*tuning*). L'addestramento del programma rispetto alla voce del lettore riduce drasticamente gli errori legati al riconoscimento.

In un secondo momento, il programma può richiedere un elenco di documenti Word o in altri formati scritti da chi parla, per memorizzare il suo lessico.

L'addestramento può proseguire quando il programma non riconosce la pronuncia di una parola, digitando il testo e registrando la pronuncia corrispondente, in modo che il programma crei un file vocale (che abbina "suono" e "scrittura" della parola in questione). Alla parola può essere associata, in alternativa, una sequenza di simboli dell'alfabeto fonetico.

Altre applicazioni

Molte applicazioni sono ideali per le soluzioni di riconoscimento vocale: operazioni di magazzinaggio e distribuzione, gestione dell'inventario, controllo qualità, lettura luce, acqua, gas, nel settore sanitario, ecc.

²⁹ Prima di utilizzare un sistema di riconoscimento vocale occorre in genere configurare il microfono e leggere al sistema alcuni brani o testi di vario genere, allo scopo di creare una serie di file di riferimento per i successivi riconoscimenti.

³⁰ In seguito analizzeremo i vari metodi statistici più in dettaglio.

³¹ In informatica, il termine macro sta ad indicare una procedura, ovvero un insieme di comandi o istruzioni, tipicamente ricorrente durante l'esecuzione di un programma.

Il riconoscimento vocale consente il multi-tasking³² che, a sua volta, comporta un aumento significativo della produttività di queste applicazioni rispetto alle tradizionali soluzioni che impegnano le mani.

Standardizzazione

Il W3C ha definito degli standard per le tecnologie vocali: il VoiceXML (dicembre 2007: versione 2.1) e il CCXML. Per la specifica di grammatiche vocali ha introdotto Speech Recognition Grammar Specification, per la sintesi vocale SSML 1.0 (Speech Synthesis Markup Language), per la pronuncia PLS 1.0 (Pronunciation Lexicon Specification), per l'interpretazione semantica dei risultati SISR 1.0 (Semantic Interpretation for Speech Recognition).

Nella mia applicazione ho utilizzato il software fornito dalla Dragon NaturallySpeaking per utilizzare la lingua italiana come lingua di interfaccia. Ovviamente per quanto riguarda la lingua inglese era disponibile, in ambiente Windows, la Speech Engine fornita dallo stesso sistema operativo (Microsoft Speech API³³) o anche altri programmi come l'IBM ViaVoce.

Dopo l'iniziale installazione attraverso un comune microfono ho predisposto il sistema operativo a ricevere il segnale audio proveniente dalla mia voce, e quindi "addestrato" il programma.

Successivamente ho costruito una semplice applicazione dotata di un *rich-text control*³⁴ in grado di funzionare come una sorta di word processor e ricevere l'input proveniente dal programma di riconoscimento vocale.



Fig. 12 - RichEdit Control in una finestra di programma.

Utilizzando il software fornito dalla Microsoft ma anche mediante altri programmi opportunamente predisposti è possibile interagire con l'applicazione, gestire la sua stessa API e avere accesso a una gamma di risorse disponibili al programma in questione.

³² In informatica, un sistema operativo con supporto per il *multitasking* (*multiprocessualità*) permette di eseguire più programmi contemporaneamente: se ad esempio viene chiesto al sistema di eseguire contemporaneamente due processi A e B, la CPU eseguirà per qualche istante il processo A, poi per qualche istante il processo B, poi tornerà ad eseguire il processo A e così via.

³³ API: in questo caso sigla per Application User Interface

³⁴ Un *rich-text control* è sostanzialmente una casella di testo con le funzionalità di un vero e proprio editor.

3. Preprocessing Tools

Una volta realizzata la trasformazione del messaggio fonico, in un messaggio testuale è possibile accedere a quest'ultimo attraverso una nuova gamma di risorse e di strumenti per l'elaborazione dei testi.

In questo capitolo presenterò alcuni algoritmi fondamentali in grado di introdurre l'argomento e permetterci di familiarizzare con programmi di semplice realizzazione e utilizzo.

I vari software sono sviluppati in Visual C++ CLI/CLR, Java o Python, ma definiti in modo tale da essere facilmente trasportabili in altri linguaggi (C# sempre per la piattaforma .NET nel cui linguaggio come già detto ho realizzato la versione del Chatbot per la rete, oppure anche JAVA).

Document Triage e Text Segmentation

Nell'analisi linguistica di un testo in forma digitale, è innanzitutto necessario definire quali sono i caratteri, le parole e le frasi presenti nel documento. Ovviamente definire tali unità comporta approcci peculiari a seconda della lingua di riferimento dei vari testi. Come sappiamo le lingue naturali contengono ambiguità a ogni livello di utilizzo e interpretazione, e i sistemi di trascrizione spesso amplificano tali ambiguità. L'attività di Text Preprocessing si può dividere generalmente in due fasi: *document triage* e *text segmentation*. Con *document triage*³⁵ intendiamo il processo di convertire un insieme di file digitali in ben definiti documenti di testo. Per i primi corpora questa attività risultava un lento processo manuale, e i corpora così costruiti non contenevano che pochi milioni di parole. Contrariamente, gli attuali corpora, realizzabili attraverso Internet, possono comprendere miliardi di parole, il che richiede un efficiente processo di elaborazione.

Questo processo può essere caratterizzato, a sua volta, da diverse fasi, in base alle origini dei file da elaborare: si ha una prima fase di *Character Encoding Identification* nella quale viene identificata la codifica digitale dei vari caratteri presenti all'interno del documento; nella seconda fase, *Language Identification*, viene determinata la lingua stessa in cui è stato scritto il documento: naturalmente le prime due fasi sono collegate vicendevolmente.

In seguito, dal testo ottenuto vengono eventualmente rimossi elementi indesiderati, come immagini, tabelle, intestazioni, vari *link* o formattazioni HTML.

A questo punto l'output dello stadio di triage è un ben definito corpus, organizzato in base alla lingua, adatto per una eventuale, ulteriore segmentazione e analisi.

Per *Text Segmentation* si intende il processo atto a convertire un ben definito corpus testuale nelle sue componenti (ad esempio parole e singole frasi).

Nella Word Segmentation si identificano all'interno del testo le varie parole o meglio i loro confini: dove cioè una parola inizia e dove la successiva finisce. Nella linguistica computazionale tali parole vengono generalmente definite come *token* e tale tipo di segmentazione conosciuto come *tokenization*. Prima della fase di *tokenization* è per altro opportuno effettuare una *normalizzazione* ("normalization") del testo, ossia stabilire, per le varie forme in cui una parola può comparire all'interno di un testo, una sua forma canonica.

³⁵ *Triage*: termine francese che indica una cernita o smistamento. Utilizzata in ambiente ospedaliero, per estensione la tecnica del *triage* viene messa in opera ogni qual volta è necessario smistare una serie di utenti che chiedono un servizio, verso gli operatori opportuni.

Ad esempio per la lingua inglese possono capitare le varie forme “Mr.”, “Mr”, “mister, o “Mister”; in italiano diverse forme possono comparire con o senza l’accento: “dopo” o “dòpo”.

Con *Sentence Segmentation* intendiamo il processo di determinare delle unità linguistiche più complesse (le cosiddette frasi) a partire dai vari token presenti nel testo. Tale attività necessita ovviamente l’identificazione dei confini tra le varie frasi, individuando quei token specifici che segnalano l’inizio di una frase e la fine di un’altra.

In pratica, le due attività sono strettamente connesse: ad esempio, per gran parte delle lingue europee, un compito essenziale è quello di individuare le parole abbreviate poiché il *punto* “.” può essere utilizzato sia per denotare una abbreviazione, sia per definire la conclusione di una frase. Nei programmi di word processing, ad esempio, esiste una funzione che se attivata, può trasformare automaticamente la lettera iniziale della parola successiva a un *punto*, in una lettera maiuscola, anche nel caso in cui il *punto* segnali un’abbreviazione.

Tokenizing, Stemming e Part of Speech Tagging Text

Ci limiteremo in questo contesto alle lingue europee e quindi ai sistemi di scrittura alfabetici di queste ultime. Ovviamente tali discorsi non potrebbero essere applicati a lingue come il *cinese* o il *giapponese* che nella forma scritta presentano forme ideogrammatiche alle quali devono essere applicati diversi criteri di segmentazione. Tralasciando il problema della codifica dei singoli caratteri alfabetici, con i vari standard ASCII e UNICODE (che assegnano una specifica configurazione di bit a un carattere: nella codifica ASCII vengono utilizzati 8 bit, cioè un byte, nella codifica UNICODE due byte) e degli altri standard per i vari alfabeti (ad esempio lo standard ISO-8859-5 presenta, per i simboli dell’alfabeto cirillico, una configurazione diversa dello standard KOI8-R) ci soffermeremo ora su altri aspetti di *preprocessing* a livello essenzialmente di singoli *token* e singole *frasi*.

Per *Text Tokenizing* si intende come già accennato, il processo di suddividere una stringa di testo in token secondo criteri prestabiliti che definiscono la vera e propria suddivisione.

Con *Stemming* indichiamo invece il processo di ricavare la radice di una parola allo scopo di permettere un più facile e complementare confronto di quest’ultima con altre parole.

Con *Part of Speech Tagging* (POS Tagging) ci riferiamo invece al processo di identificare la caratteristica grammaticale di una parola all’interno del testo, in base a una serie di predefiniti *tags*³⁶ di riferimento. Tale attività, come è facile intuire, risulta estremamente complessa poiché molte parole possono essere classificate in maniera diversa in base al contesto in cui sono utilizzate (ad esempio il token “porta” può essere un sostantivo nella frase “La porta è aperta” oppure un verbo nella frase “Il postino porta buone notizie”)

Le procedure appena descritte del resto risultano essenziali e definiscono un ottimo approccio introduttivo alle problematiche presentate. In effetti prima di elaborare qualsiasi testo è sempre necessario scomporlo nelle sue parti costitutive. Consideriamole adesso in dettaglio e osserviamone le principali caratteristiche.

³⁶ “Etichette” in grado di definire se un *token* è, ad esempio un verbo <VB>, un nome proprio <NP> ecc. Vedremo meglio il POS Tagging successivamente in questo stesso capitolo.

Tokenizing

I possibili token da prendere in considerazione sono nel nostro ambito composti da parole, numeri e simboli di punteggiatura. È essenziale definire innanzi tutto i criteri di separazione dei diversi token e quindi i simboli che determinano tale separazione. Nella mia classe di esempio *Tokenizer*, realizzata in C# per il chatbot presente in rete, ho realizzato una serie di metodi in grado di adattare tale processo alle varie esigenze. Nella routine di tokenizzazione standard (la prima qui di seguito presentata) i simboli di separazione sono assegnati a priori alla stringa *sSeparator* (“!?-_\\n”): possiamo osservare tra essi il carattere di spaziatura e il carattere di accapo ‘\\n’; nella seconda routine i simboli di separazione sono definiti negli argomenti della chiamata alla stessa routine. I vari token sono assegnati a un array di stringhe e prodotti come risultato finale *res*.

```
public static ArrayList wordsToListStand(String sSentence)
{
    ArrayList res = new ArrayList();
    String sSeparator;
    sSeparator = "!?-_\\n";
    char[] separator = sSeparator.ToCharArray();
    string[] temp = sSentence.Split(separator);
    List<String> ls = new List<String>();
    ls = temp.ToList();
    int n1;
    n1 = ls.Count;
    for (int i = 0; i < n1; i++) res.Add(ls[i].ToString());
    return res;
}
```

```
public static ArrayList wordsToListTok(String sSentence, String sSeparator)
{
    ArrayList res=new ArrayList();
    char [] separator= sSeparator.ToCharArray();
    string [] temp= sSentence.Split(separator);
    List<String> ls = new List<String>();
    ls = temp.ToList();

    int n1;
    n1 = ls.Count;
    for (int i = 0; i < n1; i++) res.Add(ls[i].ToString());

    return res;
}
```

Nella seguente routine invece scritta in Visual C++ per il chabot LUCY effettuo un adattamento della stringa di partenza rimuovendo i segni di punteggiatura dalla stessa stringa da elaborare

```
CString cleanString(CString mSentence)
{
    CString cs;
    mSentence.Remove(_T(\\' ));
    mSentence.Remove(_T(\\, ));
    mSentence.Remove(_T(\\; ));
    mSentence.Remove(_T(\\: ));
    mSentence.Remove(_T(\\? ));
    mSentence.Remove(_T(\\! ));
}
```

```

// possibilità di rimuovere gli spazi ridondanti
int len=mSentence.GetLength();
CString temp=L"";
wchar_t prevChar;
prevChar=0;
int len1;
len1=len-1;
for(int i = 0; i < len; ++i)
{
if(mSentence[i]== L' ' && i==0)continue;
if (mSentence[i]== L' ' && i==len1)continue;
if (mSentence[i]== ' ' && prevChar == ' ')
{ }
else
{
temp += mSentence[i];
prevChar =mSentence[i];
}
}
cs = temp;
return cs;
}

```

Nelle librerie di Python la tokenizzazione è implicitamente effettuata richiamando il numero d'ordine del token all'interno di un testo precedentemente acquisito.

```

>>> text5[16715:16750]
['U86', 'thats', 'why', 'something', 'like', 'gamefly', 'is', 'sò', 'good',
'because', 'you', 'can', 'actually', 'play', 'à', 'full', 'game', 'without',
'buying', 'it', 'JOIN', 'PART', 'JOIN', 'Yeah', ',', 'I', "m", 'à', 'girl',
'U98', 'n.n', 'tò', 'play', 'an', 'atmospheric']

>>> text6[1600:1700]
['We', "", 're', 'an', 'anarchò', '-', 'syndicalist', 'commune', '.', 'We',
'take', 'it', 'in', 'turns', 'tò', 'act', 'as', 'à', 'sort', 'of', 'executive',
'officer', 'for', 'the', 'week', '.', 'ARTHUR', ':', 'Yes', '.', 'DENNIS', ':',
'But', 'all', 'the', 'decision', 'of', 'that', 'officer', 'have', 'tò', 'be',
'ratified', 'at', 'à', 'special', 'bi', '-', 'weekly', 'meeting', '--', 'ARTHUR',
':', 'Yes', ',', 'I', 'see', '.', 'DENNIS', ':', 'By', 'à', 'simple', 'majority',
'in', 'the', 'case', 'of', 'purely', 'internal', 'affairs', '--', 'ARTHUR',
':', 'Be', 'quiet', '!', 'DENNIS', ':', 'But', 'by', 'à', 'twò', '-', 'thirds',
'majority', 'in', 'the', 'case', 'of', 'more', 'major', '--', 'ARTHUR', ':',
'Be', 'quiet', '!', 'I', 'order']

```

Nella fattispecie, il testo text5 risulta tratto da una chat, e text6 risulta la trascrizione del film dei Monty Python, *Monty Python and the Holy Gral*. I numeri evidenziati tra parentesi quadre (text5[16715:16750]) indicano l'indice del token da stampare.

Anche in Java, ovviamente è possibile trovare una libreria per gestire queste funzioni testuali. Il codice evidenziato segnala il tokenizer vero e proprio e i vari parametri per impostarlo. Il risultato è ovviamente un array di stringhe.

```

/*
 * utility to tokenize an input string into an Array of Strings - with a maximum
 # of returned words
 * @param s2 string containing words to tokenize
 * @param maxR maximum number of tokens to return
 * @return a List<String> of parsed tokens
 */

static public List<String> wordsToList(String s2, int maxR) {
    s2 = stripControlCharacters(s2);
    List<String> words = new ArrayList<String>();
    String x;
    int count = 0;
    try {
        StreamTokenizer str_tok = new StreamTokenizer(new StringReader(s2));
        str_tok.whitespaceChars('\'', '\"');
        str_tok.whitespaceChars('\'', '\"');
        str_tok.whitespaceChars('/', '\\');
        //str_tok.wordChars(':', ':');
        while (str_tok.nextToken() != StreamTokenizer.TT_EOF) {
            String s;
            switch (str_tok.ttype) {
                case StreamTokenizer.TT_EOL:
                    s = ""; // we will ignore this
                    break;
                case StreamTokenizer.TT_WORD:
                    s = str_tok.sval;
                    break;
                case StreamTokenizer.TT_NUMBER:
                    s = "" + (int) str_tok.nval; // .toString(); // we will
ignore this

                    break;
                default :
                    ...

            }
        } catch (Exception e) {
            e.printStackTrace();
        }
        return words;
    }
}

```

Individuazione di parole chiave

Tale attività parte dal presupposto che, nell'elaborazione di un documento per raggiungere un determinato obiettivo, non tutte le parole hanno una medesima importanza e, com'è ovvio, esistono particolari parole chiave, dotate di "importanza" e altre parole che non hanno nessun significato e che, per una più corretta analisi devono essere ignorate.

Ad esempio è spesso necessario disporre di un elenco di parole denominate *stopword* (tale elenco è definito *stoplist*) e cioè parole che non hanno alcun valore selettivo: è il caso di parole troppo comuni come congiunzioni, articoli e altro, la cui presenza nell'ambito di una elaborazione può alterare determinate considerazioni statistiche.

Ad esempio si consideri la seguente frase:

Il libro di cucina è sopra il tavolo. Il libro non mi interessa.

Il token più ricorrente è l'articolo "il" (con tre token), ma ovviamente la parola chiave del breve testo è il sostantivo "libro" che ricorre soltanto due volte. L'eliminazione di "il" da una eventuale considerazione statistica converge l'attenzione su elementi più *significativi* come "libro".³⁷

Alcune possibili *stopword* per l'italiano sono:

a
adesso
ai
al
alla
allo
allora
altre
altri
altro
anche
ancora
avere
aveva
...
...
uno
va
vai
voi
volte
vostro

Thesaurus

Prima di elaborare un testo si può far riferimento a un thesaurus di parole (una sorta dizionario di sinonimi e termini correlati a un termine di riferimento) che possono focalizzare la nostra attenzione e quella dell'elaboratore. Volendo effettuare, ad esempio, una ricerca per la parola chiave "bosco" potremo ricorrere al thesaurus e gestire tale ricerca con un insieme di parole congiunte quali "selva", "foresta", "boscaglia", "macchia", ecc., individuando quindi porzioni di testo aggiuntive attinenti alla nostra ricerca di partenza.

³⁷ Nell'ambito delle tecniche di lettura veloce viene riferito che generalmente l'80 per cento di ciò che è utile sapere è contenuto nel 20% delle parole. Utilizzando queste tecniche si fa ricorso in modo implicito a una sorta di stoplist per velocizzare la lettura e quindi l'elaborazione del testo.

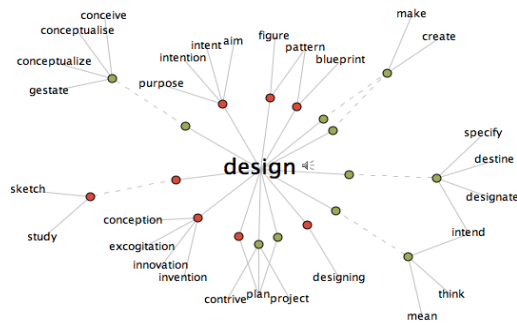


Fig. 13 - Thesaurus relativo alla parola inglese “design”

Weighting

Le parole chiave possono essere “pesate” facendo uso di informazioni statistiche sulla distribuzione dei termini. Le parole chiave possono difatti avere un significato più o meno *preciso*, e vanno dunque pesate in modo differente l’una dall’altra. Per ricondurre il concetto di *precisione* in termini matematici, sono state proposte diverse formule che a partire dalla distribuzione della parola-chiave nel singolo documento e nell’intera collezione assegnano diversi pesi a queste ultime.

Esempio pratico: assegnare automaticamente “Tags” a intere porzioni di testo.

Attraverso tale attività intendiamo definire un testo preso come riferimento con una sua categorizzazione, cioè una sorta di etichetta in grado di riferirne l’argomento essenziale come ad esempio “politica” se il testo si occupa di politica, “economia” se si occupa invece di economia. È un tipo di attività che meglio esploreremo parlando di Information Extraction ma che può costituire un ottimo esempio di utilizzo di parole chiave “pesate”.

I vari tags possibili sono definiti inizialmente all’interno di un database in *xml* e, malgrado il codice da sviluppare per tale procedura risulta relativamente semplice e non molto esteso, occorre un lavoro imponente per costruire il database delle differenti classificazioni.

Nel file *xml* di riferimento a ciascuna parola individuabile all’interno di un testo viene assegnato un punteggio “predefinito” in base a una categoria specifica. Ad esempio se incontriamo la parola “buddismo” all’interno di un testo, il punteggio della categoria *religione* assegnabile a tale testo viene incrementato di un valore relativamente alto, mentre se nello stesso testo viene incontrata la parola “intero”, conoscerà un incremento quello della categoria “computer_programming”.

File “categories.xml” con alcuni topic di esempio.

```
<tags>
  <topic name="religion_buddhism">
    <term name="buddimo" score="52" />
    <term name="buddista" score="50" />
    <term name="mente" score="50" />
    <term name="meditare" score="41" />
    <term name="buddha" score="37" />
    <term name="praticare" score="31" />
    <term name="insegnare" score="15" />
    <term name="percorso" score="14" />
    <term name="mantra" score="14" />
    <term name="pensiero" score="14" />
  </topic>
</tags>
```

```

<term name="scuola" score="13" />
<term name="zen" score="13" />
<term name="mahayana" score="13" />
<term name="soffrire" score="12" />
<term name="uomo" score="11" />
<term name="felice" score="11" />
<term name="ottenere" score="10" />
<term name="dono" score="10" />
<term name="lui" score="10" />
...
<topic>
<topic name="computers_programming">
  <term name="intero" score="128" />
  <term name="oggetto" score="120" />
  <term name="codice" score="106" />
  <term name="numero" score="79" />
  <term name="vettore" score="78" />
  <term name="stringa" score="73" />
  <term name="chiave" score="56" />
  <term name="allegato" score="49" />
  <term name="ruotare" score="47" />
  <term name="stato" score="43" />
  <term name="suono" score="42" />
...
</topic>

```

In *Java*, utilizzando una variabile hash del seguente tipo

```

private static String[] tagClassNames;
private static List<Hashtable<String, Float>> hashes =new ArrayList<Hashtable<String, Float>>();

```

è possibile produrre per ogni testo un *dizionario* di categorie e punteggi associati a tali categorie, determinabili attraverso una scansione del testo stesso.

La categoria di appartenenza di un testo verrà definita quindi in base a criteri eminentemente quantitativi e statistici: ad esempio si sceglie la categoria che ha ottenuto il maggiore punteggio.

Specificamente a partire da un testo come:

```
String s = "Il presidente si è recato in parlamento per discutere una proposta di legge riguardante la riforma fiscale"
```

si ottiene il seguente output.

```

[NameValue: news_economy : 0.91]
[NameValue: news_politics : 0.80]

```

Language Identification

È una fase fondamentale nell'ambito del Text Processing e può svolgersi secondo diversi approcci in base al tipo di lingue da identificare.

In lingue che hanno un alfabeto unico come il greco o l'ebraico si potrà identificare la lingua semplicemente stabilendo l'insieme dei caratteri utilizzati. In modo simile si potrà restringere il numero di ipotesi nello studio di lingue che condividono molti caratteri, come

ad esempio l'arabo e il persiano, il russo e l'ucraino, il norvegese e lo svedese. La differenza tra queste lingue potrà essere stabilita proprio dal fatto che una lingua può utilizzare, rispetto all'altra caratteri supplementari: il persiano ne utilizza rispetto all'arabo.

Nei casi più difficili, come nelle lingue europee, che utilizzano lo stesso insieme di caratteri ma con frequenze differenti, sarà possibile una identificazione facendo riferimento a specifiche distribuzioni di frequenza dei vari caratteri per le specifiche lingue, distribuzioni che saranno poi confrontate con quella del testo in esame, stabilendo una conclusione finale attraverso una comparazione delle diverse distanze³⁸ tra le varie distribuzioni.

Vediamo come riferimento le frequenze percentuali delle lettere più comuni tra l'italiano e l'inglese. Calcolando in un testo specifico il valore delle frequenze di tali lettere si potrebbe definire con approssimazione qual è la lingua utilizzata nel testo:

Italiano		Inglese	
E	11,79	E	12,31
A	11,74	T	9,59
I	11,28	A	8,05
O	9,83	O	7,94
N	6,88	N	7,19

Tab. 1 - Distribuzione della frequenza delle lettere per l'italiano e l'inglese. Le cifre esprimono una percentuale.

Stemming

Lo *stemming* è il processo di riduzione della forma flessa di una parola alla sua radice, detta tema. Il tema, in realtà, non corrisponde necessariamente alla radice morfologica (lemma) della parola: normalmente è sufficiente che le parole correlate siano riferite allo stesso tema (ad esempio, che *andare*, *andai*, *andò* corrispondano al tema *and*), anche se quest'ultimo non è una valida radice per la parola. Il concetto di stemming per altro non è applicabile a tutte le lingue; per esempio non è possibile ricavare la radice lessicale di parole cinesi, ma è comunque applicabile alla stragrande maggioranza delle lingue parlate.

Assumendo che la scrittura avvenga da sinistra verso destra, la radice della parola è verso la sinistra della stessa, mentre alla destra uno o più suffissi possono essere aggiunti.

All'inizio della parola è possibile trovare prefissi, ma solitamente questi alterano il significato della parola radicalmente, e quindi non vanno eliminati.

È necessaria per altro una più precisa distinzione tra *radice* e *stem* di una parola. Definiamo lo *stem* come il risultato del processo di stemming, e la radice come la parola che comprende lo stem e da cui questi etimologicamente deriva.

Funzionamento di una procedura di stemming

Possiamo separare i suffissi in tre sottoclassi, che chiameremo suffissi di tipo *a*, di tipo *i*, e di tipo *d*.

³⁸ Tale distanza potrà essere classificata con criteri statistici come lo scarto quadratico medio o altri tipi di distanza vettoriali.

Il suffisso di tipo *a* (*attached*) è una parola-particella unita alla parola principale. Nella nostra lingua, per esempio, sono suffissi di tipo *a* i due suffissi del termine “mandarglielo” (mandare + gli + lo).

Il suffisso di tipo *i* (*inflectional*) è un suffisso base della grammatica della lingua in esame, ed è applicabile a tutte le parole di un certo tipo grammaticale, con poche eccezioni. Ad esempio, in inglese il passato di un verbo si forma tipicamente aggiungendo il suffisso “ed” (con eventuali eccezioni in parte gestibili dall’algoritmo); oppure in italiano gli avverbi spesso si costruiscono posponendo “mente” al relativo aggettivo.

Il suffisso di tipo *d* (*derivational*) crea una nuova parola, spesso con un senso differente dall’originale. La possibilità o meno di unire un suffisso di questo tipo a una parola non dipende dalle regole grammaticali, ma dal dizionario. In inglese, “ness” può essere aggiunto ad alcuni aggettivi per formare i corrispondenti pronomi (kind + ness = kindness), ma ciò non vale per tutti gli aggettivi (es. big). I suffissi di tipo *d* possono variare la connotazione di una parola, per esempio in italiano il suffisso “astro” crea una forma dispregiativa (medico + astro = medicastro).

Come si può intuire, il confine tra i suffissi di tipo *i* e *d* è labile, e non è sempre siamo capaci di classificare con sicurezza un suffisso in una categoria piuttosto che nell’altra.

Normalmente i suffissi dei diversi tipi compaiono nell’ordine *d-i-a*, possiamo perciò aspettarci di rimuoverli nell’ordine inverso da destra a sinistra.

Due parole diverse che hanno lo stesso stem si definiscono *conflated* (combinare).

Ulteriori difficoltà nascono dal concetto stesso di *parola e lettera*.

Nella maggior parte delle lingue europee e derivate, una parola è una sequenza di lettere compresa tra due caratteri che non siano lettere. Ma in inglese, o in olandese, gli apostrofi, pur non essendo lettere, non sono determinanti per definire i confini di una parola.

Le lettere formano le parole, ma differenti lingue utilizzano lettere diverse, con un ulteriore confusione causata dalle lettere accentate. Per non parlare di lingue che usano alfabeti diversi, come il russo con l’alfabeto cirillico.

Infine, spesso lettere accentate, quando scritte in maiuscolo, vengono scritte con l’apostrofo anziché l’accento (*E’* anziché *È*).

Esempio di stemming dell’inglese tramite le librerie di Python: è un primo semplice approccio che individua semplicemente i suffissi indicati nella notazione tra parentesi quadre: [‘ing’, ‘ly’, ‘ed’, ‘ious’, ‘ies’, ‘ive’, ‘es’, ‘s’, ‘ment’].

```
>>> def stem(word):
    for suffix in ['ing', 'ly', 'ed', 'ious', 'ies', 'ive', 'es', 's', 'ment']:
        if word.endswith(suffix):
            return word[:-len(suffix)]
    return word

>>> raw = "This is a wonderful example distributing wisdom, for all living
being"

>>> tokens = nltk.word_tokenize(raw)
```

```
>>> [stem(t) for t in tokens]

['Thi', 'i', 'à', 'wonderful', 'example', 'distribut', 'wisdom', ',', 'for',
'all', 'liv', 'be']

>>>
```

Le varie inesattezze sono dovute all'approssimazione dell'espressione regolare definita nella routine di stemming. Si potrebbe, in alternativa, far ricorso a una versione più complessa che utilizza l'espressione regolare:

```
regex = r'^.*?(ing|ly|ed|ious|ies|ive|es|s|ment)?$'
```

e che è in grado di ottenere risultati relativamente più accurati.

Algoritmo di Porter

Uno degli algoritmi più utilizzati per lo stemming è comunque l'*algoritmo di Porter*. Tale algoritmo è dipendente dalla lingua di riferimento, e necessita di differenti formulazioni a seconda della lingua per cui è costruito: difatti il procedimento viene svolto in modo iterativo attraverso la rimozione di suffissi e prefissi che sono proprio dipendenti dalla lingua.

Gli aspetti negativi di questo algoritmo (che non utilizza un dizionario) sono soprattutto i seguenti: innanzi tutto parole differenti possono essere unificate alla stessa radice: per esempio le parole matto e mattone (questo tipo di casi, per altro, non è molto frequente). Inoltre l'algoritmo non riesce a riconoscere deviazioni morfologiche della stessa parola, per esempio vado e andiamo. Tali casi dovrebbero essere gestiti quindi attraverso una lista di eccezioni.

Vediamo in dettaglio l'algoritmo di Porter per l'italiano. La sua relativa complessità evidenzia come semplici procedure effettuate dall'uomo in maniera intuitiva una volta padroneggiata la lingua di riferimento, necessitano di approcci, per certi versi artefatti, nel caso di computer. Ho riportato per intero l'algoritmo proprio per evidenziare tale concetto: ovviamente la sua comprensione complessiva è necessaria soltanto nel caso di una *realizzazione* in una qualche routine effettivamente funzionante.

Algoritmo di Porter

Preprocessing: l'italiano può avere le seguenti vocali accentate: *á é í ó ú à è ì ò ù*. Per prima cosa, bisogna sostituire tutti gli accenti acuti con quelli gravi, inoltre trasformare in maiuscolo le **u** dopo la **q**, e le **u** e le **i** che si trovano tra vocali.

Nell'esempio:

inquietudine diventa inqUietudine

annoiato diventa annoIato

A questo punto le vocali saranno: *a e i o u à è ì ò ù*

Definiamo le variabili R1, R2 e RV di una parola.

R1 è la regione dopo la prima non-vocale che segue una vocale, oppure è uguale a una regione nulla alla fine della parola, se non ci sono non-vocali.

R2 è la regione dopo la prima non-vocale che segue una vocale in R1, oppure è uguale alla regione nulla alla fine della parola se non ci sono consonanti.

(la definizione di vocale cambia da lingua a lingua. In francese, *è*, è una vocale, in italiano, la *i* tra vocali non è una vocale, da questo si comprende la sostituzione precedente).

Vediamo un esempio di R1 e R2, per una parola inglese:

```
b   e   a   u   t   i   f   u   l
                        |<----->|   R1
                        |<----->|   R2
```

La lettera t è la prima non vocale che segue una vocale, quindi R1 è iful. In iful, la prima lettere che segue la vocale è f, quindi, R2 è ul.

RV è definita come segue: se la seconda lettera è una consonante, RV è la regione dopo la vocale successiva, o se le prime due lettere sono vocali, RV è la regione dopo la successiva consonante, altrimenti RV è la regione dopo la terza lettera. Se non è possibile trovare queste posizioni allora RV è la fine della parola.

Continuiamo con le regole di sostituzione:

Effettuare sempre lo step 0 e lo step 1.

Step 0:

Cercare nella parola il più lungo dei seguenti suffissi:

*ci gli la le li lo mi ne si ti vi sene gliela gliele glieli glielo gliene mela mele meli me
lo mene tela tele teli telo tene ceta cele celi celo cene vela vele veli velo vene*

che seguono uno tra

(a) *and endo*

(b) *ar er ir*

in RV. Nel caso di (a) il suffisso è eliminato, nel caso di (b) è sostituito da e. (*guardandogli* à *guardando*,
accomodarci à *accomodare*)

Step 1: Rimozione dei suffissi standard:

Cercare il più lungo suffisso tra quelli seguenti ed effettuare le azioni indicate:

*anza anze ico ici ica ice iche ichi ismo ismi abile abili ibile ibili ista iste isti istà istè
istì oso osi osa ose mente atrice atrici ante anti*

eliminarle se in *R2*

azione azioni atore atori rimuoverle se in *R2*

Se precedute da *ic*, rimuoverle se in *R2*

logia logie

sostituire con *log* se in *R2*

uzione uzioni usione usioni

sostituire con *u* se in *R2*

enza enze

sostituire con *ente* se in *R2*

amento amenti imento imenti

eliminarle se in *RV*

amente

rimuoverle se in *R1*

se precedute da *iv*, rimuoverle se in *R2* (and ancora, preceduto da *at*, rimuoverlo se in *R2*), altrimenti,

se preceduto da *os*, *ic* or *abil*, rimuoverle se in *R2*

ità

rimuoverle se in *R2*

se preceduti da *abil*, *ic* or *iv*, rimuoverli se in *R2*

ivo ivi iva ive

rimuoverli se in *R2*

se preceduti da *at*, rimuoverli se in *R2* (e se ancora preceduti da *ic*, rimuoverli se in *R2*)

Effettuare lo step 2 se nessun suffisso è stato rimosso dallo step 1.

Step 2: Suffissi di verbi

Cercare il più lungo suffisso tra quelli seguenti in *RV* e se trovato eliminarlo.

ammo ando ano are arono asse assero assi assimo ata ate ati ato ava avamo avano avate avi avo emmo enda ende endi endo erà erai eranno ere erebbe erebbero erei eremmo

eremo ereste eresti erete erò erono essero ete eva evamo evano evate evi evo Yamo iamo immo irà irai iranno ire irebbe irebbero irei iremmo iremo ireste iresti irete irò irono is

*ca iscano isce isci isco iscono issero ita ite iti ito iva ivamo ivano ivate ivi ivo ono uta
ute uti uto ar ir*

Effettuare sempre lo step 3 a e 3b.

Step 3a

Rimuovere le finali **a, e, i, o, à, è, ì** or **ò** se non sono in *RV*, e la precedente **i** se non è in *RV* (*crocchi -> crocch, crocchio -> crocch*)

Step 3b

Sostituire le **ch** (or **gh**) con **c** (or **g**) se in *RV* (*crocch -> crocc*)

Infine,

ripristinare le **I** e le **U** in minuscolo.

Esiste in rete una libreria utile per lo stemming: Oleander Stemming Library, scritta in C++, rilasciata con licenza BSD (Berkeley Software Distribution) e disponibile in una varietà di lingue.

Esempio di stemming realizzato con l'algoritmo di Porter

Abbandonata	abbandon
abbandonate	abbandon
abbandonati	abbandon
abbandonato	abbandon
abbandonava	abbandon
abbandonerà	abbandon
abbandoneranno	abbandon
abbandonerò	abbandon
abbandono	abband
abbandonò	abbandon
abbaruffato	abbaruff
abbassamento	abbass
abbassando	abbass
abbassandola	abbass
abbassandole	abbass
abbassar	abbass
abbassare	abbass
abbassarono	abbass
abbassarsi	abbass
abbassassero	abbass
...	...
pronto	pront
pronuncerà	pronunc
pronuncia	pronunc
pronunciamento	pronunc
pronunciare	pronunc
pronunciarsi	pronunc
pronunciata	pronunc
pronunciate	pronunc
pronunciato	pronunc
pronunzia	pronunz
pronunziano	pronunz
pronunziare	pronunz
pronunziarle	pronunz
pronunziato	pronunz
pronunzio	pronunz
pronunziò	pronunz
propaga	propag
propagamento	propag

Tab. 2 - Esempio di stemming di parole italiane.

Pos Tagging

Con *part-of-speech tagging* si intende comunemente un livello di analisi linguistica del testo che assegna a ciascuna unità linguistica sintatticamente autonoma (o token), individuata all'interno di un testo, una definizione in base ai suoi tratti morfosintattiche a

partire da un insieme di categorie di riferimento. Questo insieme contiene generalmente l'indicazione del *lemma* (o forma lessicale di citazione), la sua *categoria grammaticale* (o parte del discorso) e i *tratti morfologici* ad essa associati. Mentre l'insieme dei tratti morfologici può variare significativamente da uno schema di annotazione a un altro (e da una lingua a un'altra), praticamente tutti gli schemi di annotazione morfo-sintattica forniscono l'indicazione della categoria grammaticale. Il livello di categorizzazione morfo-sintattica si applica in generale all'output di un analizzatore morfologico, individuando, tra le possibili analisi morfologiche di una parola omografa, quella appropriata al suo contesto.

id	form	lemma	pos	feats
1	Il	il	RD	MS
2	danno	danno;dare	S;V	MS;P3IP
3	non	non	B	NULL

↓

id	form	lemma	pos	feats
1	Il	il	RD	MS
2	danno	danno	S	MS
3	non	non	B	NULL

Fig. 10 - Esempio di categorizzazione.

A scopo esemplificativo ho utilizzato il Pos tagger del Penn Treebank³⁹ attraverso un Wrapper⁴⁰ in C++ che ne semplifica il funzionamento. Le annotazioni per il tagging sfrutta le annotazioni per l'italiano fornite da Marco Baroni. L'elenco dei Tags utilizzati risulta tratto dalla seguente lista (riprodotta parzialmente, ma presente in Appendice):

ADJ
adjective (e.g. ROSSI, BELLISSIMA)

ADJ:abr
adjectival abbreviation (ON.)

ADV
adverb (e.g. SIMPATICAMENTE, NON)

...

ART
article (e.g. IL, LA)

...

NOUN
common name (e.g.. TAVOLO, GIOCO)

³⁹ Il Penn Treebank Project annota testi scritti in base alla loro struttura linguistica. In particolare vengono prodotti dei *parser* (sorta di programmi) in grado di produrre informazioni sintattiche e semantiche riguardanti qualsiasi testo in forma digitale. Uno dei siti riferimento è quello dell'Università della Pennsylvania: <http://www.upenn.edu/>

⁴⁰ Un programma che in genere permette di utilizzare più agevolmente un altro programma.

...

WH

wh element (e.g. DOVE, COME, PERCHE')

WH:che

che as a wh element (e.g. L'UOMO CHE HAI VISTO, HAI DETTO CHE)

Un esempio di POS TAGGING nella lingua italiana è il seguente:

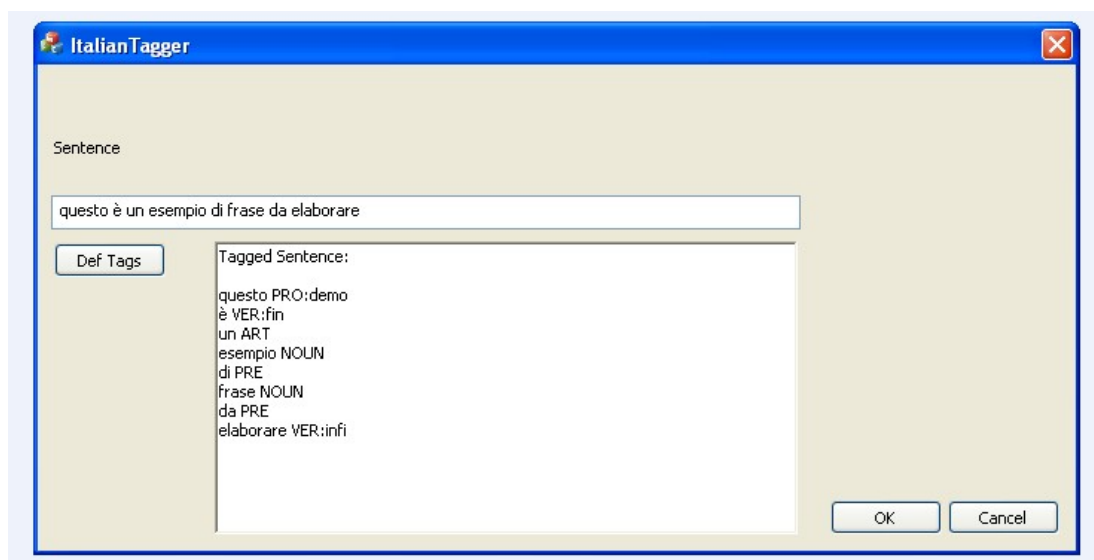


Fig. 11 - Schermata del programma ASItalianTagger

Esistono in ogni caso numerosi algoritmi che si occupano di tale problema e che possono differire in base alla lingua di riferimento. Allo stato dell'arte prevalgono sostanzialmente gli algoritmi basati sul modello markoviano nascosto (HMM)⁴¹ che, ancora una volta, a partire da criteri in prevalenza statistici, riescono a produrre il tagging del testo in base ai criteri specificati dai tag a disposizione.

Nell'ambito della costruzione di una interfaccia dialogante il riferimento a una possibile definizione della frase secondo i vari *tag* presupposti può risultare di notevole ausilio ad esempio nella costruzione di un set di domande, a partire da un determinato testo.⁴²

⁴¹ Un *processo stocastico markoviano* o *processo di Markov* è un processo stocastico nel quale la probabilità di transizione che determina il passaggio da uno stato di sistema, all'altro, dipende unicamente dallo stato di sistema immediatamente precedente (proprietà di Markov) e non dal *come* si è giunti a tale stato (in quest'ultima ipotesi si parla di *processo non markoviano*).

Tale processo prende il nome dal matematico russo Andrej Andreevič Markov che per primo ne sviluppò la teoria.

Un *Modello di Markov nascosto* (*Hidden Markov Model* - HMM) è una catena di Markov i cui vari stati non sono osservabili direttamente. I modelli nascosti di Markov sono conosciuti in modo particolare proprio per le loro applicazioni nel riconoscimento dello schema temporale dei discorsi parlati e della scrittura a mano.

⁴² Riconoscere un nome proprio, può permettere la costruzione di domande del tipo "Chi...?", "Quale personaggio...?", ecc.

Text Clustering

Per *Text Clustering* intendiamo l'attività di raggruppare tra loro testi simili (in base a determinate caratteristiche).

È una attività propria dell'Information Retrieval a cui accenneremo brevemente in questo capitolo per dare un'idea di come l'attività di *preprocessing* possa adattarsi alle più svariate esigenze.

Per svolgere un semplice Text Clustering sono di norma utilizzati i seguenti procedimenti

- 1) calcolare l'intersezione delle parole comuni tra due documenti
- 2) calcolare l'intersezione delle parole comuni considerandone la radice.
- 3) calcolare l'intersezione di tag assegnati al documento
- 4) calcolare l'intersezione dei nomi propri di persona e di luoghi presenti nel documento.

Nel mio algoritmo considererò soltanto i primi due criteri: il semplice programma da me utilizzato, valido a scopo dimostrativo tratto dal testo di Mark Watson, *Practical Artificial Intelligence Programming with Java*⁴³, confronta tra loro una serie di testi in inglese di cui due a carattere economico e gli altri due generici articoli di cronaca. Il risultato ottenuto dal confronto è il seguente:

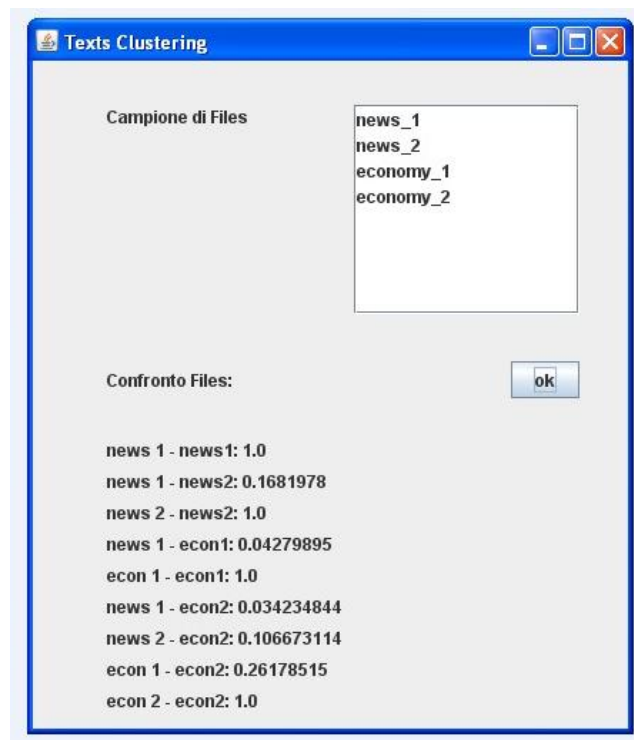


Fig. 12 - Esempio di Texts Clustering.

Il valore del confronto è compreso tra “0” e “1”. Come si può osservare un file rapportato a se stesso genera un valore pari a “1”, mentre è più alto il valore del rapporto tra due testi di genere affine.

⁴³ Watson Mark, *Practical Artificial Intelligence Programming with Java* (2001). Il libro è scaricabile all'indirizzo: <https://leanpub.com/javaai>

4. Information Extraction

Dopo questa breve panoramica su alcuni degli strumenti fondamentali per l'elaborazione di un testo, è necessario, a mio avviso, trattare in modo più esaustivo quella disciplina che è alla base di qualsiasi forma di Text Processing, l'*Information Extraction*.

Per Information Extraction si intende il compito di estrarre informazioni da dati "privi di struttura". Nel nostro contesto, per dato privo di struttura intendiamo un *dato* digitale le cui informazioni, pur possedendo una loro coerenza, sono codificate in modo difficile da elaborare (per un computer). Testi scritti e parlati, immagini, registrazioni video e audio sono tutte forme di dati privi di struttura.

Nel nostro caso, la capacità di estrarre informazioni a partire da un testo è alla base di qualsivoglia, ulteriore, sviluppo nell'ambito dell'intelligenza artificiale applicata all'elaborazione del linguaggio naturale. Senza informazioni iniziali su cui far riferimento è, del resto, difficile anche solo presupporre la nascita di una intelligenza o di un sistema che a sua volta sviluppi o *generi* strutture testuali.

E, tanto meno, realizzare una interfaccia dialogante in grado di rispondere a domande specifiche.

Descriverò in questo capitolo le teorie, gli algoritmi e le soluzioni più importanti realizzate, nel corso degli ultimi decenni, allo scopo di individuare porzioni di informazione in un testo definito come dato di ingresso.

Importanza della Information Extraction

Si considerino i seguenti esempi: una compagnia vorrebbe conoscere le opinioni riguardanti un prodotto realizzato di recente esaminando i contenuti di alcuni Blog presenti nel Web.

Un'altra compagnia si propone di utilizzare analoghe informazioni per costruire un dettagliato resoconto riguardante lo sviluppo di determinate tecnologie nell'ambito della produzione di materiale elettronico.

O ancora: un'agenzia per la ricerca nello spazio mette a disposizione dei propri astronauti una vasta quantità di documenti tecnici da consultare prima di una determinata missione.

In altri contesti: il governo di un determinato paese sta raccogliendo dati riguardo eventuali disastri naturali che possono verificarsi all'interno del proprio territorio: lo scopo è di istruire i servizi di emergenza con un rapporto dettagliato e riassuntivo dei dati presi in esame.

Un'agenzia di intelligence sta investigando le attività di gruppi terroristici in tutto il mondo. Ha a disposizione un ricco database e milioni di nuovi contributi informativi, resi disponibili sotto le più svariate forme di memorizzazione: gli analisti di tale agenzia vorrebbero costruire un dettagliato rapporto di tutti gli eventi terroristici in una particolare regione geografica nel corso degli ultimi anni.

In un contesto molto più vicino alla nostra vita quotidiana uno studente di legge è interessato a esaminare le decisioni della magistratura riguardo casi di divorzio già passati in giudizio: ha a disposizione migliaia di precedenti risoluzioni maturate nei vari tribunali e dovrebbe consultarle integralmente per produrre delle conclusioni a proposito.

In ambito scientifico, un gruppo di ricerca nel campo biomedico sta sperimentando nuovi trattamenti farmaceutici e desidera conoscere tutte le possibili maniere con le quali uno specifico gruppo di proteine può interagire con altre proteine e che cosa può risultare da

questo tipo di interazioni: il gruppo ha a propria disposizione decine di migliaia di articoli, resoconti di conferenze e rapporti tecnici.

Gli esempi appena considerati hanno numerosi elementi in comune:

- 1) innanzitutto la richiesta di informazioni;
- 2) la risposta a tale richiesta è di solito presente all'interno di elementi informativi come testi e immagini;
- 3) è impossibile per un essere umano elaborare tutti questi dati poiché essi sono semplicemente troppi: e occorrerebbe una quantità di tempo esponenziale per poterli consultare tutti.
- 4) i computer non sono in grado di esaudire direttamente le richieste secondo metodologie di uso comune poiché, come abbiamo appena detto, le informazioni sono presenti in forma non strutturata.

L'*Information Extraction* è l'ambito dell'NLP che si propone quindi di risolvere tali problematiche.

Cenni storici

L'idea di realizzare un'estrazione automatica di informazioni a partire da determinati testi è da ricondursi inizialmente agli studi di Zellig Harris⁴⁴ che nel 1950 propose, tra i primi, l'eventualità di identificare le principali strutture semantiche di un testo e automaticamente estrarle da quest'ultimo allo scopo di ulteriori elaborazioni. Gli studi di Harris condussero Naomi Sager⁴⁵ a considerare un approccio analogo a partire da testi scientifici o documenti in ambito medico: già alla fine degli anni ottanta esistevano gruppi di studiosi in grado di sviluppare applicazioni relativamente semplici, per l'estrazione di riferimenti dai quotidiani e dai documenti militari.

Allo scopo di valutare sistematicamente i raggiungimenti conseguiti in questo campo dell'NLP la marina militare americana, organizzò una serie di sette conferenze, le *Message Understanding Conferences* (MUC) a partire dal 1987 fino al 1998. I lavori nel campo dell'IE e i compiti assegnati a tale disciplina furono perciò largamente influenzati dai propositi e dalle indicazioni di tali incontri. Ad esempio nella prima conferenza si diede grande rilievo al riconoscimento di eventi nell'ambito temporale, nella sesta si sottolineò l'importanza della "name extraction", nella settima venne introdotto il concetto di "template relation"

Generalmente, all'interno di tali conferenze venivano predefiniti dei template⁴⁶ riguardanti un campo ristretto di informazione (come ad esempio le attività terroristiche nel Sudamerica) e si richiedeva ai diversi sistemi di riempire i cosiddetti *slot* (gli spazi vuoti corrispondenti ai template) con le informazioni richieste.

⁴⁴ Zellig Sabbetai Harris (1909 - 1992): rinomato linguista e scienziato americano. È conosciuto per i suoi lavori nell'ambito della linguistica strutturalista e nell'analisi del discorso e per la scoperta della natura trasformazionale del linguaggio.

⁴⁵ Naomi Sager (n. 1927) risulta una delle principali responsabili del Linguistic String Project, un progetto di elaborazione del linguaggio naturale basato sull'analisi trasformazionale, e la teoria linguistica delle stringhe.

⁴⁶ Il termine inglese *template* (letteralmente "sagoma" o "calco") in informatica indica un documento o programma nel quale, come in un foglio semicompilato cartaceo, su una struttura generica o standard, vengono definiti spazi temporaneamente "bianchi" da riempire successivamente. In questo ambito, la parola è traducibile in italiano come "modello", "semicompilato", "schema", "struttura base", "ossatura generale" o "scheletro", o più correntemente "modulo".

I presupposti che consentono lo sviluppo di una tale disciplina.

In generale il tentativo di estrarre informazioni da un testo attraverso algoritmi specifici, utilizzabili da un computer trova la propria ragione d'essere nell'organizzazione linguistica insita nei vari testi. Qualsiasi testo, in qualsivoglia lingua, consiste difatti in una complessa struttura di modelli e configurazioni (*pattern*) ricorrenti e iterativi, che formano un costrutto coerente e significativo. Questa caratteristica è una conseguenza del cosiddetto principio di *composizionalità*, un concetto della filosofia linguistica che teorizza l'esistenza di molteplici possibili approcci al linguaggio, e sottolinea, sostanzialmente, come il significato di ogni complessa espressione linguistica risulta una funzione dei significati delle sue parti costituenti.

Ad esempio, nella lingua italiana è possibile riconoscere la costante e ricorsiva presenza di un soggetto, di un verbo, di una serie di complementi.

Il loro significato individuale, e la maniera con cui sono ordinati nel testo, ci permettono di determinare e ipotizzare ciò che una frase o un intero testo vogliono esprimere. Difatti, al di fuori di regole specifiche e di convenzioni usate in maniera ripetitiva, sarebbe praticamente impossibile interagire in maniera funzionale e "automatica" con qualsiasi forma di messaggio.

Non è ancora stato univocamente stabilito come questi diversi componenti si influenzino esattamente, ma secondo molte teorie linguistiche a cui certi aspetti dell'NLP fanno riferimento, è possibile parlare di "catena realizzazionale" (*realizational chain*) una nozione che ha radici molto antiche: ad esempio nei lavori del linguista indiano Panini nel sesto secolo a.C.

Secondo queste teorie, il significato di un testo prodotto in una qualsiasi lingua è realizzato nella sua forma superficiale, dall'interazione di una quantità di ulteriori livelli linguistici, ciascuno dei quali effettua la proiezione di determinate proprietà, dai livelli più astratti verso il livello superficiale. In Panini, il significato di una frase inizia a partire da una idea nella mente del parlante o di chi scrive. L'idea viene quindi elaborata in uno stadio in cui vengono formati i primi concetti semantici tradotti a loro volta, in un livello successivo, in un insieme di concetti grammaticali e lessicali. Che verranno quindi tradotti nella sequenza di caratteri o di foni propri della lingua utilizzata.

L'Information Extraction (e in genere l'intero ambito dell'NLP) parte dal presupposto che il processo a cui abbiamo accennato risulti, in un certo senso, bidirezionale e che le idee, i concetti contenuti in un messaggio, in un testo linguistico, siano recuperabili a partire dalle varie realizzazioni di superficie e attraverso una serie di procedimenti inversi.

In altre parole, l'Information Extraction presuppone che, anche se le informazioni semantiche in un testo e la loro organizzazione linguistica non sono *immediatamente* trasparenti, dal punto di vista computazionale possono essere recuperate considerando la struttura superficiale del testo, la sua regolarità, i suoi *pattern* ricorrenti, e in genere tutte quelle caratteristiche sfruttabili da un algoritmo per individuare una porzione di significato e di informazioni utili a una successiva elaborazione.

Da cosa nasce la necessità di efficienti modelli di IE.

Come abbiamo più volte sottolineato la nostra realtà è caratterizzata da una sovrabbondanza esponenziale di informazioni. Nessuno è in grado, attualmente, di fare una valutazione precisa della quantità di informazioni disponibili in forma digitale ma secondo una stima del 2002 la quantità di nuovi dati digitali (soprattutto questi ultimi) prodotta

soltanto per quell'anno ammontava a circa 5 exabytes,⁴⁷ una entità corrispondente a circa 9500 miliardi di libri o grande 500.000 volte l'intera Biblioteca del Congresso (che si suppone contenga 10 terabytes di informazione).

Secondo i vari calcoli, il Web di "superficie" conteneva circa 167 terabyte di informazione, e secondo altre indicazioni il Web "profondo" (Deep Web: ossia la porzione del web memorizzata su database e accessibile agli esseri umani, ma non indicizzabile automaticamente dai motori di ricerca) era circa 400 o 500 volte più ampia. Esistevano circa 66800 terabyte di dati, in gran parte nella forma di testi, immagini, registrazioni video e audio. Una rappresentazione grafica dell'ordine di grandezza di questi valori numerici può essere la seguente.

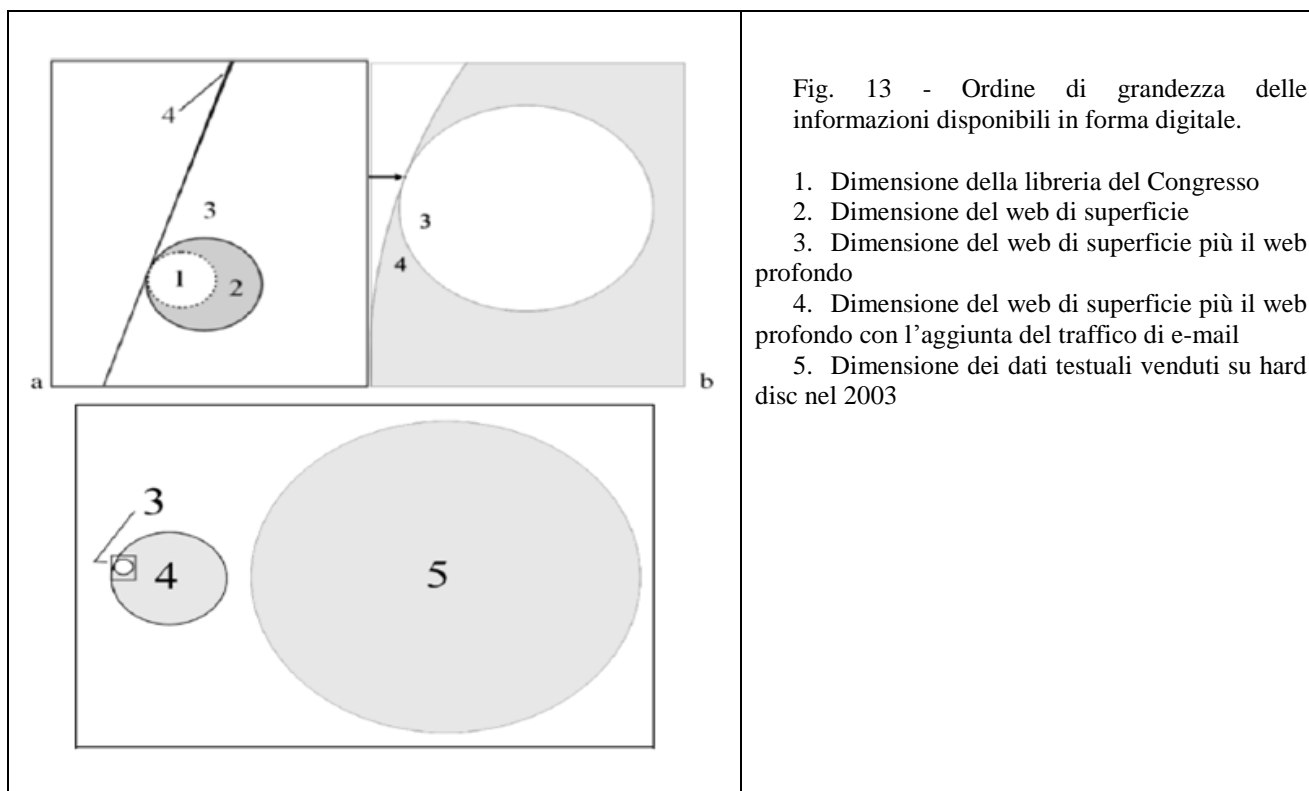


Fig. 13 - Ordine di grandezza delle informazioni disponibili in forma digitale.

1. Dimensione della libreria del Congresso
2. Dimensione del web di superficie
3. Dimensione del web di superficie più il web profondo
4. Dimensione del web di superficie più il web profondo con l'aggiunta del traffico di e-mail
5. Dimensione dei dati testuali venduti su hard disc nel 2003

Le applicazioni di *Information Retrieval* e *Information Extraction* cercano perciò di contribuire, alla gestione di questa sovrabbondanza di dati e risolvere in parte le problematiche prodotte da tale *pantagruelica* realtà.

Ma cosa intendiamo più esattamente con *Information Retrieval* e *Information Extraction*?

⁴⁷ Il byte è tipicamente formato da otto bit e il termine fu coniato da Werner Bucholz nel luglio del 1956 all'inizio del progetto del computer IBM Stretch. Agli inizi degli anni 1960 esistevano due tipi di computer: *scientifici* e *commerciali*. Nei computer scientifici, per la rappresentazione dei numeri decimali, venivano usati 4 bit (ad esempio 5 = 0101, 9 = 1001, per un totale di sedici configurazioni possibili), mentre per i computer commerciali venivano usati 6 bit (64 configurazioni) per rappresentare il *set grafico* stampabile, che comprendeva 26 caratteri alfabetici (solo maiuscole), 10 numeri, e da 11 a 25 simboli speciali (+ - * / > ecc.). I vari multipli del byte sono il kilobyte (10^3 byte) il megabyte (10^6) il gigabyte (10^9) il terabyte (10^{12}) il petabyte (10^{15}) l'exabyte (10^{18}) lo zettabyte (10^{21}) e lo yottabyte (10^{24})

L'*Information Retrieval* consiste nell'attività di individuare, a partire da una specifica richiesta informativa e da un insieme di documenti messi a disposizione, un sottoinsieme di questi documenti tra i quali l'informazione richiesta è o può essere presente.

Allo scopo di svolgere tale ricerca in maniera efficace i testi disponibili per la consultazione vengono "indicizzati" in anticipo (classificati attraverso una sorta di operazione di catalogo più o meno complessa), come nella costruzione di un catalogo in una biblioteca): la ricerca viene effettuata consultando, appunto, i riferimenti predeterminati. Le tecniche di indicizzazione devono, per altro, essere in grado di rendere le eventuali ricerche il più possibile flessibili alle richieste di un qualsiasi utente, (come si verifica, oggi, nei motori di ricerca più celebri). Gli utenti che ne usufruiscono difatti possono produrre le più svariate richieste informative: di conseguenza è necessario realizzare sistemi quanto più flessibili anche se, talvolta a scapito, come vedremo, dell'accuratezza e della precisione o rilevanza, con cui i risultati di una ricerca vengono prodotti.

In ogni caso, a motivo della crescita esponenziale delle informazioni a nostra disposizione, questo approccio, ispirato come è evidente ai paradigmi della tradizionale organizzazione di una biblioteca cartacea, non risulta più così efficiente.

In poche parole, quando la quantità di documenti utili per i nostri propositi risulta esorbitante la consultazione degli stessi risulta inevitabilmente dispendiosa in termini di tempo e di accuratezza. A questo proposito le tecniche di *Information Extraction* di cui parleremo ora più espressamente si propongono di gestire tale crescente inefficienza. I documenti ritenuti significativi allo scopo di una ricerca vengono a loro volta esaminati automaticamente allo scopo di definire meglio la loro rilevanza e il loro contenuto. Nell'ambito delle interfacce vocaliche e più in generale dei sistemi di risposta a domande (*question answering systems*), l'*Information Extraction* risulta basilare poiché in questi sistemi l'utente finale pone una specifica domanda a cui verrà fornita una risposta più o meno specifica, non solo individuando la plausibile risposta corrispondente a una eventuale richiesta, ma anche esprimendo quest'ultima corredata di una opportuna analisi e classificazione dei suoi contenuti. Si cerca cioè di produrre informazioni e ulteriori informazioni connesse a queste ultime, in grado di permetterne una migliore interpretazione.

In tale senso possiamo parlare di una elementare e primitiva forma di comprensione del linguaggio (Natural Language Understanding) che utilizza competenze di carattere morfologico, sintattico e semantico.

In aggiunta alle situazioni di cui abbiamo parlato, l'*Information Extraction* gioca un ruolo importante anche nell'ambito del recupero di informazioni in testi diversi dalla lingua in cui viene effettuata la richiesta informativa (Cross-Language Information Retrieval, o CLIR). In quest'ambito è necessario distinguere i concetti semantici in genere indipendenti dal linguaggio, da quegli elementi lessicali che invece caratterizzano una determinata lingua. Si cerca cioè di realizzare un modello indipendente dalla lingua di utilizzo che, per certi versi, presenta molte analogie con il presupposto teorico dell'esistenza di una interlingua in grado di ospitare qualsiasi traduzione e che a sua volta, quindi, può risultare lingua di partenza per la traduzione conclusiva nella lingua obiettivo. Questo presupposto viene sfruttato ad esempio in alcuni algoritmi utili alla traduzione automatica dei testi (Machine Translation).

Esempi concreti di Information Extraction:

Name Entity Recognition (NER), Name Extraction

Gran parte dei testi con cui siamo soliti interagire contengono nomi propri, di persona, di luoghi, e in genere di svariate entità nominali, (società, organizzazioni, ecc.).

Di conseguenza uno dei compiti iniziali e forse più immediati della IE risulta proprio quello di individuare tali nomi e scoprire se appunto essi appartengono a persone, organizzazioni, luoghi o altre entità.

La Name Extraction è particolarmente importante in diverse discipline, come ad esempio nelle applicazioni mediche, e in genere ovunque sia necessaria una terminologia ben definita e il più possibile priva di ambiguità.

Esempi.

I primi sistemi per realizzare l'estrazione di nomi propri erano basati su una struttura di regole annotate manualmente. Tali regole consistevano in espressioni regolari che, quando erano presenti nel testo, etichettavano la porzione di testo corrispondente, come una entità "nome".

Si consideri la seguente regola:

[token che inizia con maiuscola]+ "Soc. Coop."+ → Organizzazione

Una serie di termini con un token in maiuscola iniziale seguiti dai token "Soc. Coop." possono essere etichettati come "*Organizzazione*": ad esempio Bonanza Soc. Coop.

Nella seguente notazione invece, i termini che iniziano con maiuscola e che seguono il token "signor" sono definibili come nomi propri di persona (*persona*).

"signor"+ [lettera maiuscola? token che inizia con maiuscola] → Persona

Il simbolo '+' indica una o più istanze del primo termine (Coop. Alfa Beta) mentre nel secondo esempio il simbolo '?' indica l'opzionalità del termine precedente (il signor A. Bianchi). Nel verificarsi delle circostanze definite nelle regole, i termini tra parentesi quadre vengono considerati come nomi propri.

Signor Rossi → persona

Signor J. R. Hewin → persona

Soc. Coop. Logistica Industriale → Organizzazione

Altre regole utilizzabili con questo approccio sfruttano la ricorrenza di nomi propri piuttosto consueti, presenti possibilmente in un elenco di cui siamo a disposizione:

nome proprio comune? token che inizia per maiuscola → persona

In riferimento a tale regola, dalla sequenza "Carlo Tavola", dopo aver constatato che la stringa Carlo è un nome, individuamo in "Tavola" un ulteriore nome proprio, un cognome, e non un nome comune.

L'elenco di nomi preso in considerazione può essere costruito a partire da informazioni raccolte nelle maniere più svariate: ad esempio estraendo i nomi statisticamente più comuni

in un censimento della popolazione, da liste di persone molto conosciute, compagnie, e luoghi (non solamente relativi alla realtà presente): informazioni disponibili su Wikipedia o in qualsiasi altra fonte che presenti una lista e una classificazione analoga (ad esempio siti in cui sono indicati i nomi più frequenti in una determinata lingua di una determinata nazione).

Algoritmo *ItProperName*

In questo esempio concreto di algoritmo di estrazione ho costruito un file *propername.list* contenente un insieme di nomi propri italiani. Questo file è letto nel costruttore di classe `Names` e serializzato in una struttura `HashMap`⁴⁸.

All'input di una frase come la seguente:

Giorgio Rossi aprirà a Roma una nuova succursale della Nokia...

dovrà seguire un output con le seguenti caratteristiche

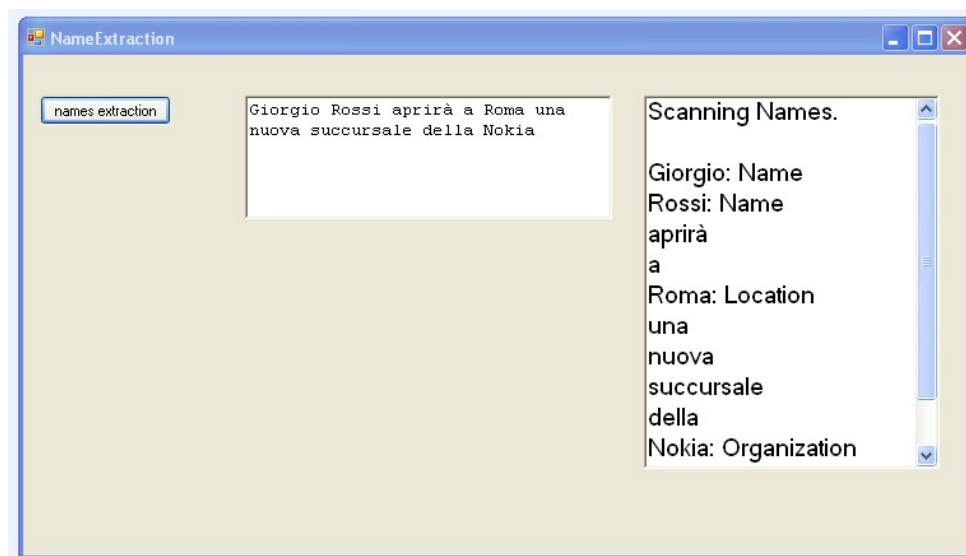


Fig. 14 - Esempio di Name Extraction

La classificazione verrà utilizzata dopo aver costruito tre dictionary che contengono nomi propri di luoghi e di organizzazioni.

```
names=this->buildDictionary(L"names");
locations=this->buildDictionary(L"locations");
organizations=this->buildDictionary(L"organizations");
```

...

```
if(names->ContainsKey(sToken)) sRow= sToken +": Name";
if(locations->ContainsKey(sToken)) sRow = sToken +": Location";
if(organizations->ContainsKey(sToken)) sRow = sToken +": Organization";
```

⁴⁸ In informatica una *hash table*, detta anche *hash map*, (in italiano *tabella hash*) è una struttura dati usata per mettere in corrispondenza, in maniera univoca, una data *chiave* con un dato *valore*.

Ovviamente le procedure appena descritte e altre analoghe ad esse possono essere utilizzate attraverso più scansioni dello stesso testo e successive iterazioni. Tale approccio si basa sulla eventualità che molti nomi possono apparire più di una volta in un documento.

Alcune istanze di tali nomi possono cioè presentarsi in contesti che permettono al nome stesso di essere classificato, mentre altre istanze possono trovarsi in un contesto che non permette tale identificazione; ripetendo più volte il processo è possibile identificare entrambe le istanze dello stesso nome, e procedere successivamente a individuare altri nomi legati a esse. Per esempio:

Franco Amato ha reso noto un suo imminente viaggio a Roma per motivi di lavoro. Il signor Amato è del parere che...

Il programma di analisi può essere inizialmente incerto se considerare “Franco Amato” come il nome di una ditta o di una persona o addirittura come una combinazione di un aggettivo e un participio (“franco” e “amato”). In ogni caso, una volta incontrato il sintagma “signor Amato”, attraverso una successiva scansione del testo, l’algoritmo potrà etichettare “Franco”, e quindi “Franco Amato”, come il nome proprio di una persona, con sufficiente certezza. Tale procedimento può essere esteso a un intero insieme di documenti. Le regole di annotazione manuale presentate, sono notevolmente precise ma richiedono una considerevole competenza testuale, abilità nella loro individuazione e configurazione, e naturalmente dispendio di tempo.

Altre forme di Extraction

Event Extraction

L’*Event Extraction* risulta tra i primi tipi di applicazioni per l’estrazione di informazioni a essere stato studiato. A partire da uno specifico tipo di eventi (“attacco terroristico”, “disastro aereo”, “acquisizione di una società in borsa”), si cerca di identificare istanze di quell’evento e, per ogni istanza, definire i suoi argomenti e modificatori.

Nella terminologia del MUC, la caratterizzazione di questo tipo di evento venne chiamata *scenario*, e l’insieme di slot da riempire, *scenario template*.

Un caso speciale di Event Extraction si verifica quando sappiamo in anticipo che un documento contiene esattamente una *istanza* di quell’evento, e il nostro compito è appunto quello di individuare i suoi argomenti e modificatori.

Possiamo includere tra questi eventi l’instestazione di un discorso pubblico: il nostro obiettivo sarà determinare il nome dell’eventuale oratore, il titolo del discorso, l’orario e il luogo in cui il discorso è o verrà effettuato. Ma possiamo anche considerare la redazione di un curriculum vitae, nella quale il nostro scopo è quello di estrarre specifiche informazioni biografiche; o, infine, molto importante, la classificazione di materiale pubblicitario.

Il compito appena descritto è generalmente definito Implicit Relation Extraction (IRE) e ha notevoli somiglianze con il Name Tagging per lo sfruttamento del modello sequenziale, ma con la semplificazione (in molti casi) dovuta all’esistenza di una sola istanza da cercare in ogni documento.

Esiste una vasta gamma di studi riguardanti l’IRE e molteplici corpora utilizzabili per testare le varie applicazioni.

Come nel caso della Name Extraction, i primi sistemi di Event Extraction erano basati su regole e “patterns” specificati manualmente.

I costrutti identificanti i vari eventi potevano includere una quantità di modificatori applicabili agli argomenti centrali, e perciò si rendeva necessario un certo grado di analisi sintattica, se non altro sufficiente a individuare il soggetto e il complemento di alcuni verbi principali; i pattern venivano identificati in termini di queste relazioni sintattiche. Per esempio, il pattern:

NP(organizzazione) licenzia NP(persona)

con il quale si cerca di individuare il licenziamento del dipendente di una società, funziona rintracciando la parola chiave “licenzia” e quindi individuando con NP(Organizzazione) il nome di una organizzazione, o un pronome, o un riferimento all’organizzazione stessa e con NP(persona) un nome proprio di persona, una frase nominale o un sintagma indicante una classe di persone.

Per esempio:

La FIAT licenzia gli operai del comparto verniciatura.

In riferimento alla frase precedente sarà possibile costruire una relazione strutturabile come in (a) in base alle varie corrispondenze individuate.

(a) NP(org)1 licenzia NP(person)2 → event(org: 1, person: 2, position: -, action: licenzia)

o anche

(b) Le dimissioni di NP(person)1 in qualità di NP(position)2 → event(org: -, person: 1, position: 2, action: licenzia)

Un tale procedimento può risultare non del tutto soddisfacente poiché può comportare il riempimento dei vari “template” con valori, e stringhe di caratteri privi di informazione (nel caso di un riempimento con pronomi “egli” o varie forme di epiteti e altri costrutti come “il candidato”, “il suddetto”) anche se i vari nomi specifici sono stati menzionati altrove nel documento.

Ovviamente, è sempre necessario individuare le varie entità univocamente, e definirle in tali schemi nella loro univocità, allo scopo di poter sfruttare e recuperare l’informazione raccolta in qualsiasi momento.

Problemi nell’individuazione di entità nominali

I nomi propri sono delle espressioni (*referring expression*), che si riferiscono a entità esistenti nel mondo reale. Uno sviluppo naturale della *name extraction* o *name tagging* consiste nel trovare ogni *referring expression* (nomi, frasi nominali, pronomi, e identificare quelle che appartengono alla stessa entità).

Tale “entity extraction” o estrazione di entità comprende:

1. l’identificazione e la classificazione di tutte le frasi che riferiscono a entità di specifiche tipologie semantiche; e cioè nomi, frasi nominali, e pronomi. Ci riferiamo a queste categorie come *entity mentions*.

2. il collegamento a tutte le *entity mentions* riferite alla stessa entità.

Identificare le frasi nominali è un compito standard nell’analisi sintattica. Per ogni frase nominale, determiniamo la sua estensione e la sua *testa* sintattica.

Si esamini la frase “Il poliziotto catturò il ladro a mezzanotte”. All’interno di questa frase si possono individuare diversi gruppi di parole. Per vedere quali sono questi gruppi, possiamo osservare che “a mezzanotte” è, per così dire, costruito intorno a una preposizione, ossia “a”, mentre “il poliziotto” è costruito intorno a un nome, “poliziotto”. Notiamo subito che intorno all’elemento centrale della costruzione si possono realizzare molti altri gruppi simili: intorno ad “a”, “a casa”, “a tarda sera”, “a Parigi”; intorno a “poliziotto”, “un poliziotto”, “quel poliziotto”, “il poliziotto di quartiere”, ecc. L’elemento che svolge la funzione di “a” in “a mezzanotte” o di “poliziotto” in “il poliziotto” è chiamato *testa* del gruppo di parole: i gruppi come “a mezzanotte”, visto che la loro *testa* è una preposizione, sono detti gruppi (o sintagmi) preposizionali (SP); i gruppi come “il poliziotto”, la cui *testa* è un nome, sono chiamati gruppi (o sintagmi) nominali (SN). Altri tipi o gruppi di parole sono i sintagmi verbali (SV) e i sintagmi aggettivali (SA): nei SV la *testa* è un verbo, nei SA la *testa* è un aggettivo.

Nella frase “*Il famoso linguista di Roma ama la lirica*”, la prima frase nominale ha estensione “Il famoso linguista di Roma” e *testa* “linguista”. Nel caso di frasi nominali la cui *testa* è un nome, il tipo semantico della frase è il tipo del nome, come determinato dal procedimento di estrazione dei nomi. Il problema di collegare insieme le varie *entity mentions* definisce il cosiddetto problema dell’*anaphora resolution*.⁴⁹

Esempio:

Il celebre filosofo, Carlo Bianchi, si è recato quest’oggi a Roma. Nella città eterna è stato invitato per tenere una conferenza.

Nella *fattispecie*: “Carlo Bianchi” e “il celebre filosofo”, sono coreferenti. “Roma” e “città eterna” sono anch’essi coreferenti. A seconda della rispettiva collocazione all’interno della frase, possiamo parlare di referenze anaforiche (“Il celebre filosofo”) e “cataforiche” (città eterna).

Alcuni tipi di relazioni di primaria importanza

Come abbiamo sottolineato, nel contesto dell’estraneazione delle informazioni, una *relazione* rappresenta un rapporto di relazione tra due entità; una *relation mention* è un’espressione di questa relazione, e coinvolge due *entity mention*.

Tra le relazioni più importanti possiamo evidenziare quelle riguardanti:

1) un luogo (permanente o temporaneo)

Giorgio Bianchi che attualmente vive a Roma
Lo stabilimento FIAT di Torino

2) cittadinanza o origine (tra una persona e una nazione)

Lisa Del Papa nata a Berlino
Il famoso bardo inglese

⁴⁹ *Anaphora Resolution*: è il procedimento per individuare le diverse anafore presenti in un testo e le rispettive corrispondenze.

3) appartenenza (tra una persona e una organizzazione)

Il presidente della *FIAT* - *President* (FIAT, X)

4) relazioni di parentela

Il *fratello* di Francesco - *Fratello* (Francesco, X)

Semantic Role Recognition

Tale attività riguarda il tentativo di assegnare funzioni semantiche ai costituenti sintattici di una frase. Ci si interessa a determinate azioni o stati, o i vari partecipanti ad esse e alle loro definizioni circostanziali. Ovviamente la definizione di tali ruoli semantici può essere effettuata con criteri spesse volte molto differenti a secondo delle teorie prese in riferimento.

Esempio:

Lo spettatore batté le mani dall'entusiasmo.

Lo spettatore: agente
Le mani: parte del corpo
Entusiasmo: causa

Time Line Recognition

Tra i parametri di maggiore importanza durante l'estrazione di informazioni da un testo, troviamo i cosiddetti parametri temporali. Essi indicano espressioni temporali di carattere assoluto come ad esempio: il 14 luglio 1789, l'autunno del 1492, mezzogiorno; o espressioni relative ("ieri", "la scorsa settimana", "il prossimo secolo"). Di notevole importanza sono le espressioni che indicano una durata di tempo ("un'ora", "dieci giorni"); tutte queste espressioni ovviamente possono anche essere connesse tra di loro e interagire ("due giorni prima della partenza") o anche costituire un insieme ("ogni giorno").

Tra le *relazioni temporali* possiamo individuare le seguenti: *X prima di Y*, *X uguale a Y*, *X durante Y* e altre ancora. Il procedimento di ricostruire una cosiddetta *timeline*, per quanto apparentemente semplice e intuitivo dal punto di vista di un essere umano, necessita di sofisticati ragionamenti, e implica il ricorso ad algoritmi di una notevole complessità per un computer.

Esempio:

Lo scorso 16 aprile 2013 ho superato il mio ultimo esame. Per tre mesi avevo dovuto studiare moltissimo.

Le indicazioni temporali presenti nella frase possono essere schematizzate dal punto di vista della *timeline* nella seguente maniera:

16 gennaio 2013 → 16 aprile 2013: studio.
16 aprile del 1013: esame.

Regole codificate manualmente e metodi supervisionati.

Come è evidente dagli esempi a cui abbiamo appena accennato, la maggior parte delle relazioni (sebbene non tutte) è espressa, attraverso parole di una singola frase, che si trovano a una distanza relativamente corta tra di loro.

Ciò rende fattibile la possibilità di creare dei *pattern* manualmente, e individuare molte di queste relazioni. Suggerisce anche un approccio per creare una sorta di tagger relazionale e cioè impostare un classificatore che gestisca tutte le entità che hanno un rapporto di vicinanza, con un output che potrebbe descrivere la presenza della relazione oppure la sua assenza.

Al di là delle relazioni a breve distanza, è facile però verificare come una classificazione basata esclusivamente su sequenze di parole analoghe non è sempre adeguata: possono verificarsi delle frasi nelle quali argomentazioni circostanziali possono essere aggiunte e fraposte tra le due entità che vogliamo prendere in considerazione.

Ad esempio nella frase:

Gennaro ha aperto un chiosco di gelati a Napoli.

La relazione “*siTrovaIn*” che dovrebbe avere come argomenti i token “Gennaro” e “Napoli” è costruita con i due termini agli estremi opposti della frase.

Simili esempi determinano la necessità di ricorrere a strutture ad albero, o ad altre con analoghe caratteristiche, per definire il tipo di connessione esistente tra le varie parole che compongono una frase. In questa nuova dimensione spaziale sarebbe possibile costruire la relazione *siTrovaIn* come un arco che congiunge i due nodi Gennaro e Napoli.



Fig. 15 - Esempio di albero di dipendenza

Ulteriori numerose analoghe casistiche possono essere evidenziate con estrema facilità.

Si necessita come illustrato, di molteplici livelli di rappresentazione, in grado di ridurre l'impatto degli errori, piuttosto consueti nell'ambito dell'IE, nell'analisi sintattica

Metodi parzialmente supervisionati e non supervisionati

Come per l'estrazione di nomi propri, il costo delle annotazioni manuali, in termini di tempo e precisione, e la disponibilità di una notevole quantità di testi da annotare hanno condotto a una serie di ricerche, per individuare metodi di estrazione di relazioni senza l'ausilio di alcuna supervisione.

Il modo di procedere è in un certo senso sempre lo stesso. Partiamo da entità e relazioni già conosciute, come ad esempio relazioni tra il nome di un autore, il titolo di un suo libro ([Herman Melville, *Moby Dick*], [J. R. R. Tolkien, *The Hobbit*], ecc.). e procediamo nella scansione di un vasto corpo di documenti, raccogliendo le varie sequenze di parole che intervengono all'interno di questi piccoli contesti, allo scopo di creare la relazione intesa. Cerchiamo poi sempre nello stesso corpus, altre entità che corrispondono al pattern individuato, rilevando perciò nuove relazioni: le nuove coppie che appaiono attraverso questo processo saranno aggiunte all'insieme originale e quindi lo stesso processo verrà ulteriormente ripetuto.

Ad esempio nella frase:

Lo scrittore di *Moby Dick*, Herman Melville

si può ricostruire il pattern "Lo scrittore di [T], [A]" dove per [T] intendiamo il titolo dell'opera e per [A] il suo autore. Quindi, se nell'ambito di un testo incontriamo una frase come

Lo scrittore di *Tom Sawyer*, Mark Twain

siamo in grado di individuare il pattern ricorrente e ricostruire la relazione [Mark Twain, *Tom Sawyer*].

Come per i metodi supervisionati, tali procedimenti variano a seconda di come i diversi *patterns* sono caratterizzati, anche se il criterio principale è come spesso accade, quello di utilizzare metodi che favoriscono la velocità di scansione, in modo da analizzare la maggiore quantità di testi possibili.

Naturalmente la discriminante essenziale in questa attività, risulta nella scelta dei *patterns* che spesse volte possono risultare troppo generali e ambigui. Per esempio come abbiamo visto, un pattern utilizzabile allo scopo di individuare la relazione tra il titolo di un libro e il suo autore è legato alla presenza di una virgola " ," tra le due entità; ma una virgola ovviamente può avere altri significati ed essere utilizzata per altri scopi. Altri modi di procedere possono anch'essi trarre giovamento da determinate regolarità della lingua.

Se una istanza di una sequenza di token è un nome di un particolare tipo, altre istanze della stessa sequenza molto probabilmente rappresentano un nome dello stesso tipo. Possiamo trarre vantaggio da questa proprietà per creare un programma in grado di individuare solo un piccolo insieme di nomi.

Come già in parte anticipato, supponiamo che all'inizio si disponga di pochi indicativi contesti nominali; per esempio, che " "signor" o " "signora" siano seguiti da un nome proprio di persona, e che un nome terminante con la notazione "Soc. Coop.", per esempio, sia il nome di una organizzazione. Con un approccio simile a quello che abbiamo descritto in precedenza, viene utilizzato un vasto corpus non ancora classificato e etichettato e procediamo con il tagging di tutte le istanze nominali che corrispondono ai pattern appena

menzionati. Quindi “etichettiamo” le altre istanze che comprendono i nomi appena estratti. In seguito esaminiamo l’insieme dei nuovi nomi etichettati per riscontrare se esistono dei contesti (oltre a quelli iniziali) che possono essere associati in modo consistente con nomi di un particolare tipo. Se ciò si verifica, aggiungiamo tali nomi ai termini iniziali che individuano un contesto e ripetiamo il procedimento.

Per esempio, se dopo una prima elaborazione individuiamo il termine Smith come un nome di persona e effettuando una successiva elaborazione rileviamo che esso compare nel contesto “presidente Smith”, possiamo sfruttare il termine “presidente” come una nuova entità nominale in grado di definire “un contesto” e nell’ambito di una successiva, ulteriore elaborazione qualora si incontri il sintagma “presidente Stone”, il termine Stone verrà annotato come nome proprio.

Altri pattern presi in considerazione possono avere differenti caratteristiche: tra i più validi riscontriamo pattern costruiti sulle similitudini o analoghe forme di comparazione; nella lingua italiana abbiamo *X* cioè *Y*. Se troviamo il sintagma ‘Il celebre poeta’ equiparato a ‘Leopardi’ o ‘Dante’ (es: Il celebre poeta, cioè Leopardi), l’istanza *Y* (Leopardi), come ho già detto, potrà essere considerata un nuovo riferimento per individuare nuovi pattern e ulteriori nomi.

Proponiamo uno schema riassuntivo che indica i diversi tipi di Information Extraction appena elencati:

<i>Information Extraction Task</i>	<i>Extraction Unit</i>	<i>Linguistic Context</i>	<i>Goal</i>
Named entity recognition	Parole Gruppi di parole	Frase Testo	Riconoscimento di una entità
Coreference resolution di frasi nominali	Parole Gruppi di parole	Frase Testo Molteplici testi	Riconoscimento di una entità
Riconoscimento ruolo semantico	Parole Gruppi di parole	Frase	Riconoscimento frase
Riconoscimento di relazioni fra entità	Parole Gruppi di parole	Frase Testo Molteplici testi	Discorso Comprensione della storia
Timeline Extraction	Parole Gruppi di parole	Frase Testo Molteplici testi	Discorso Comprensione della storia

Tab. 2 - Esempi di Information Extraction, le unità correlate all’estrazione, i contesti linguistici e gli eventuali scopi dell’Information Extraction

L’approccio sequenziale

In sintesi è opportuno ribadire come molte delle attività riguardanti l’estrazione di informazioni si basano su informazioni anch’esse già estratte automaticamente. Viene

spesso effettuata, cioè, una analisi bottom up,⁵⁰ impostata su diversi stadi; ad esempio in riferimento alla frase:

Giorgio Bianchi si recò a Roma in compagnia di Carla Rossi: rimase insieme a lei per dieci giorni

è necessario innanzitutto individuare i nomi Giorgio Bianchi e Carla Rossi per risolvere l'anafora relativa al pronome "lei" e quindi procedere oltre.

Una rappresentazione di tale approccio può risultare nel seguente schema:

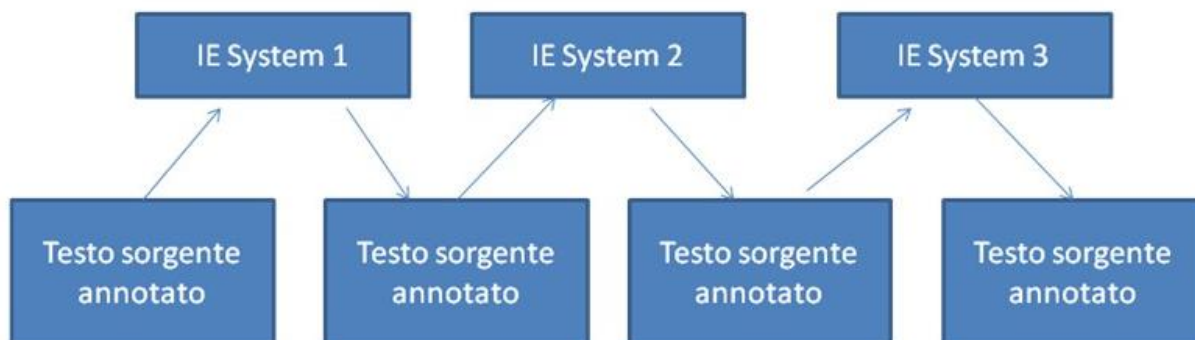


Fig. 16 - Un sistema di Information Extraction a "cascata"

Organizzazione delle informazioni raccolte

Come abbiamo ampiamente evidenziato l'Information Extraction è molto più di una semplice estrazione dati: si propone anche una loro organizzazione (come nei casi di riconoscimento di nomi congiuntamente a eventi in cui tali nomi hanno "preso parte").

Si faccia, per ulteriore chiarificazione, riferimento alle seguenti frasi:

La Microsoft ha acquisito la Powerset.
La Powerset è stata acquisita dalla Microsoft.

Si potrebbe riconoscere che non solo "Microsoft" e "Powerset" risultano grandi compagnie, ma che secondo quanto espresso dalle frasi si è verificato un particolare evento, e cioè l'acquisizione di una compagnia da parte di un'altra: nella fattispecie la prima Microsoft acquista l'altra compagnia.

Microsoft → Organizzazione
Ha acquisito → Relazione
Powerset → Organizzazione

Altro esempio:

Giorgio Bianchi lavora per la FIAT

⁵⁰ I modelli *top-down* e *bottom-up* (*dall'alto verso il basso* e *dal basso verso l'alto*, rispettivamente) sono strategie di elaborazione dell'informazione e di gestione delle conoscenze, riguardanti principalmente il software e, per estensione, altre teorie umanistiche e teorie dei sistemi.

Giorgio Bianchi → Persona
Lavora per → Relazione
FIAT → Nome di società

Come si può dedurre dagli esempi presentati, in queste procedure è opportuno individuare le informazioni memorizzandole in forme facilmente accessibili per una elaborazione automatica, e in grado di consentire la definizione di tali relazioni.

Assegnare uno specifico tag a ogni token di una sequenza è un procedimento denominato *sequence tagging task*.

Generalmente è possibile ricorrere a strutture di markup basate sull'XML, come nel seguente esempio. La frase:

Il signor Bianchi lavora per la Fiat di Torino

È etichettabile come:

Il signor <name type="person">Bianchi</name><relation> lavora per </relation> la
<name type="organization">Fiat</name> di
<name type="location">Torino</name>.⁵¹

Parimenti:

Il signor Giorgio Rossi è il nuovo amministratore della Deutsche Bank a Roma

diventa:

Il signor<name type="persona">Giorgi Rossi</name> <relation> è il nuovo amministratore </relation> of
<name type="organizzazione">Deutsche Bank</name> of <name type="luogo">Roma</name>.

Il linguaggio di markup Xml, che sfrutterò ampiamente nella progettazione della interfaccia dialogante, è estremamente funzionale allo scopo che ci siamo prefissi, ma altri linguaggi analoghi potrebbero essere utilizzati.

Per le storie tratte dai quotidiani, questo approccio è sostanzialmente standard, ma in base al contesto esistono strutture e elenchi di categorie (indicate nell'attributo "type" dell'esempio: type="organizzazione" - type="luogo") molto più ricche, con gerarchie che prevedono anche più di 200 nomi collegati all'attributo in questione. Domini diversi includeranno ovviamente diverse tipologie di nomi; testi riguardanti la genetica, per esempio, comprenderanno nomi di geni e proteine (type="proteina").

Va comunque ribadito che qualsiasi tipo di estrazione di informazioni viene spesso volte effettuato all'interno di un contesto specifico, dove determinate informazioni hanno rilevanza e dove invece informazioni di carattere diverso possono essere trascurate. Tale approccio riflette, in un certo senso, quello che noi esseri umani abbiamo nei confronti della realtà, quando quotidianamente cancelliamo e ignoriamo le informazioni provenienti dal mondo circostante per concentrarci invece su quelle di rilevante importanza nel contesto al quale siamo interessati.

⁵¹ Un esempio analogo, e la notazione ad esso relativa sono stati introdotti, all'interno delle valutazioni del MUC-6.

In ogni caso, organizzare le informazioni estratte da un determinato testo (abbiamo accennato alla funzionalità del linguaggio *Xml*) in una struttura semantica che serva come modello astratto degli eventi riguardanti il mondo reale, risulta piuttosto complessa.

Un esempio di testo annotato secondo strutture di markup, analoghe a quelle appena descritte, può essere il seguente tratto da un documento del MUC-6 redatto nel 1995.

```
<HL> <ENAMEX TYPE="ORGANIZATION">Eastern Air</ENAMEX> Proposes
Date For Talks on Pay-Cut Plan</HL>
<DD> <TIMEX TYPE="DATE">01/23/87</TIMEX></DD>
<SO> WALL STREET JOURNAL (J)</SO>
<IN> LABOR TEX AIRLINES (AIR) </IN>
<DATELINE> <ENAMEX TYPE="LOCATION">MIAMI</ENAMEX> </DATELINE>
<TXT>
<p>
<s> <ENAMEX TYPE="ORGANIZATION">Eastern Airlines</ENAMEX> execu-
tives notified union leaders that the carrier wishes to discuss
selective wage reductions on <TIMEX TYPE="DATE">Feb. 3</TIMEX>.
</s>
</p>
<p>
<s> Union representatives who could be reached said they hadn't
decided whether they would respond. </s>
</p>
<p>
<s> By proposing a meeting date, <ENAMEX
TYPE="ORGANIZATION">Eastern</ENAMEX> moved one step closer to-
ward reopening current high-cost contract agreements with its
unions. </s>
<s> The proposal to meet followed an announcement <TIMEX
TYPE="DATE">Wednesday</TIMEX> in which <ENAMEX
TYPE="PERSON">Philip Bakes</ENAMEX>, <ENAMEX
TYPE="ORGANIZATION">Eastern</ENAMEX>'s president, laid out pro-
posals to cut wages selectively an average of <NUMEX
TYPE="PERCENT">29%</NUMEX>. </s>
<s> The airline's three major labor unions, whose contracts
don't expire until year's end at the earliest, have vowed to re-
sist the cuts. </s>
</p>
```

Fig. 17 - Annotazione di una frase dal MUC-6 Documento N. 870123-0009

Si può osservare sin dalle prime righe come venga annotata la ditta Eastern Airlines (attraverso il markup “ENAMEX” con attributo TYPE=“ORGANIZATION”), oppure a metà del documento l’annotazione della espressione temporale “Feb. 3” (<TIMEX TYPE=DATE >Feb. 3</TIMEX>)

Una simile struttura dell’informazione testuale consentirebbe di per sé una gestione dei dati da parte di una interfaccia dialogante.

Ad esempio per rispondere alla domanda: “quali sono le ditte menzionate nel testo?” basterebbe “scorrere” il testo annotato e recuperare le informazioni contrassegnate all’interno del markup <ENAMEX></ENAMEX> e con l’attributo “ORGANIZATION”.

Il problema, come abbiamo più volte fatto osservare, si presenta ovviamente nel caso di testi non annotati.

Ma quali convenzioni è possibile adottare per rendere tali annotazioni funzionali per qualsiasi sistema rivolto a interpretarle?

Esistono molteplici possibili approcci: uno dei più funzionali consiste nell’assegnare a ciascun token un tag utilizzando la convenzione BIO.

In tale convenzione, il primo token di una entità nominale è classificato come B-PERSON mentre il secondo come I-PERSON. Per nomi di organizzazioni si procede in modo analogo indicando con B-ORG il primo token, con I-ORG il secondo token.

Esempio di tokenizing attraverso la convenzione BIO

Mr -
Harry B-PERSON
Hoople I-PERSON
was -
named -
CEO -
of -
Harry's B-ORG
Hogs I-ORG
of -
San B-LOCATION
Francisco I-LOCATION

Esempi di annotazioni, concettualizzazioni, rappresentazioni

Roger C. Schank: Conceptual Dependency Theory⁵²

Schank parte dal presupposto che esista una base comune e interlinguistica con la quale è possibile definire le strutture specifiche dei concetti espressi all'interno di una determinata lingua.

Queste strutture o *concettualizzazioni* sono composte di concetti primari la cui interconnessione è governata da un insieme ristretto di regole sintattiche e un più ampio insieme di regole semantiche. Schank distingue quattro classi di concetti primari, definiti *ruoli*:

- 1) Picture Producers (PP) : rappresentano gli oggetti fisici veri e propri
- 2) Acts (ACT) : rappresentano le azioni primitive
- 3) Picture Aiders (PA) : modificano i PP e di solito rappresentano uno stato di questi ultimi attraverso dei valori specifici (attributi)
- 4) Action Aiders (AA) : modificano gli ACT.

A questi ruoli se ne aggiungono altri tra i quali ad esempio

- 1) Locations (LOC) definizioni di luogo.
- 2) Times (T) : definizioni di tempo.

Il nucleo e al tempo stesso la parte più contestata del CDT è definibile nell'insieme degli undici ACT che teoricamente dovrebbero essere in grado di rappresentare (nelle loro combinazioni), qualsiasi azione un agente può svolgere in una realtà fisica.

Per esempio uno degli ACT usato più di frequente è definito come PTRANS ed esprime un cambio di luogo di una entità fisica (PP) nel mondo preso in considerazione.

In CDT la frase:

Giorgio va a Milano

⁵² Roger Schank (n. 1946) è uno studioso americano di Intelligenza Artificiale, imprenditore ed esperto in psicologia cognitiva. La sua homepage è all'indirizzo: <http://www.rogerschank.com/>

potrà essere rappresentata in uno schema come il seguente:

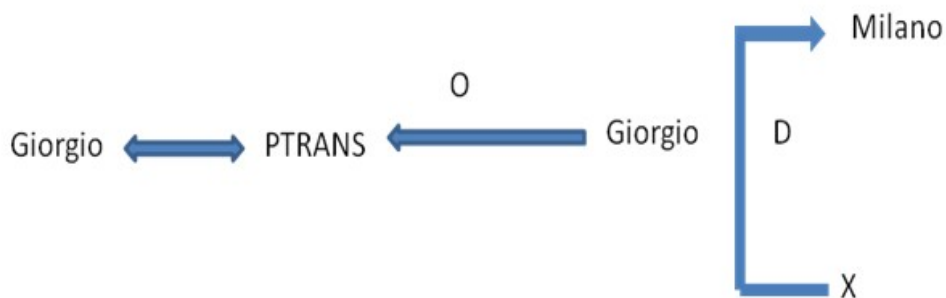


Fig. 18 Rappresentazione CDT della frase “Giorgio va a Milano”

Nel diagramma la doppia freccia identifica una relazione tra ACT e una entità fisica (l’entità che esegue l’ACT), la lettera “O” una relazione oggettiva e “D” una relazione direzionale. L’intero costrutto indica che esiste un oggetto fisico “Giorgio” che esegue un “ACT” per il quale l’oggetto “Giorgio” cambia la propria ubicazione: il luogo in cui si trova (indicato con X) diventa nella fattispecie Milano.

Una rappresentazione della frase:

Giorgio si trasferisce da Roma a Milano

Risulterebbe quindi la seguente:

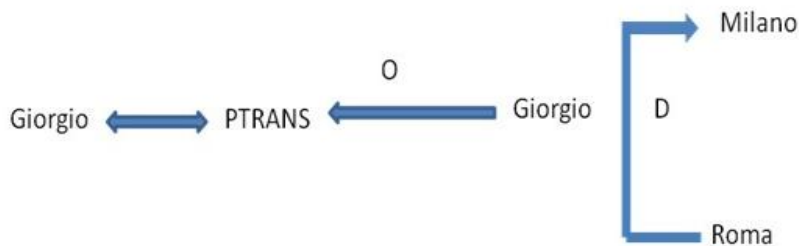


Fig. 19 - Rappresentazione CDT della frase “Giorgio si trasferisce da Roma a Milano”

I concetti possono formare combinazioni in accordo con un insieme di regole concettuali sintattiche, sottoposte a delle restrizioni *formulative* in base ai loro argomenti.

Utilizzando il framework appena descritto è possibile ricostruire rappresentazioni concettuali di azioni individuali e di stati.

Per completare questo modello formale, Schank ha introdotto un altro insieme di regole definite *causal syntax rules*, che rendono possibile la costruzione di semplici catene di eventi prendendo in considerazione differenti tipologie di relazioni causali. La maniera di

rappresentare azioni e loro causalità può essere sintetizzare in una serie di *script*: nella tabella 2 è rappresentato uno script dell'attività "prendere l'autobus."

Esempi di ACT PTRANS

X1 PTRANS X1 da LOC1 a LOC2	Una entità X1 cambia la propria posizione da LOC1 a LOC2. Ad esempio una persona X1 si reca da una posizione sconosciuta (LOC1) alla fermata dell'autobus (LOC2)
X2 PTRANS O1 da LOC3 a LOC2	Una persona (X2) e cioè l'autista dell'autobus sposta la posizione dell'autobus (O1) da un luogo LOC3 alla fermata dell'autobus LOC2. (Teoricamente dovrebbe essere definita una condizione secondo la quale l'entità O1 è un veicolo ed è possibile per gli esseri umani muovere tali veicoli
X1 PTRANS X1 da LOC2 a O1	La persona X1 presso la stazione degli autobus LOC2 sale nell'autobus O1
X1 ATRANS O2 da X1 a X2	La persona X1 paga una tariffa in denaro (O2) all'autista dell'autobus (X2). ATRANS è usato per esprimere il trasferimento del possesso di una entità. In questo caso il denaro.
X2 ATRANS O3 da X2 a X1	L'autista X2 consegna il biglietto (O3) alla persona X1
...	...
X2 PTRANS O1 da LOC2 a LOC4	L'autista X2 conduce l'autobus O1 dalla stazione LOC2 a un altro luogo non specificato
X1 PTRANS X1 da O1 a LOC4	X1 scende dall'autobus (O1) presso la fermata nel luogo LOC4

Tab. 3 - Esempio di uno script semplificato: *prendere l'autobus*

Un altro degli esempi classici di questa teoria, analogo al precedente è lo script del *ristorante*, che ha le seguenti caratteristiche

Scena 1 Ingresso

S PTRANS S nel ristorante
 S ATTEND osserva i tavoli
 S MBUILD dove sedere
 S PTRANS S al tavolo
 S MOVE S in posizione di seduto

Scena 2 Ordinazione

S PTRANS menu (S prende il menu dal tavolo)
 S MBUILD scelta del cibo
 S MTRANS segnale al cameriere
 Cameriere PTRANS al tavolo
 S MTRANS "desidererei ordinare" al cameriere

Il cameriere PTRANS in cucina

Scena 3 Mangiare

Il cuoco ATRANS cibo al cameriere

Il cameriere PTRANS da S

Il cameriere ATRANS denaro al cliente per il conto

S ATRANS denaro al cameriere

S PTRANS fuori dal ristorante

...

Ci sono naturalmente molte possibili varianti di questo script che hanno a che fare con differenti tipi di procedure e piccoli avvenimenti verificabili in un ristorante. Ad esempio nella modalità del pagamento del conto, o nella consegna dell'ordinazione in cucina. Tali varianti presentano quindi delle occasioni di fraintendimento e comunque determinano una possibile, non corretta, rappresentazione della realtà.

Ciò nonostante, per via della sua struttura e organizzazione piuttosto dettagliata il modello di Schank risulta di notevole utilità per applicazioni di Information Extraction. Gli script possono essere facilmente memorizzati ed elaborati, riducendo i tempi di calcolo. È possibile, con essi, realizzare delle inferenze logiche e consentire la predizione di concettualizzazioni a partire dalle concettualizzazioni date. Nel modello della CDT si cerca cioè di predire e prevedere tutti gli eventi che si possono verificare in una data situazione.

Le concettualizzazioni possono essere rappresentate in script della forma:

$(\text{role}^1 (\text{var}_1, \dots, \text{var}_n) \text{ACT} (\text{var}_1, \dots, \text{var}_p) \text{role}^2 (\text{var}_1, \dots, \text{var}_q), \dots, \text{role}^k (\text{var}_1, \dots, \text{var}_r))$

La struttura definisce una regola sintattica, e le variabili, regole semantiche.

Naturalmente, malgrado queste premesse, come spesso accade ai modelli formali che si ripropongono di sintetizzare il linguaggio naturale umano e la sua capacità di rappresentare la realtà (il modello di Schank è soltanto uno tra tanti esempi), numerose critiche possono essere mosse nei confronti di questo modello.

Innanzitutto il fatto che risulti applicabile all'interno di domini ristretti (dove funziona in modo eccellente): ma oltre un certo livello di complessità gli script risultano insufficienti per esprimere il meccanismo di anticipazione che sembra caratterizzare l'approccio *intelligente* degli esseri umani alla realtà; è praticamente e anche teoricamente impossibile costruire uno script per ogni situazione concepibile, anche se molte situazioni possono essere gestite con un sufficiente numero di previsioni in grado di rendere il modello funzionante dal punto di vista eminentemente pratico. Per esempio l'atto "prendere posto a sedere" (nella scena 1) è descritto in modo sommario senza porre attenzione alle molteplicità di varianti con le quali è possibile compiere lo stesso atto e realizzare lo stesso evento.

Inoltre, tale modello non è in grado di valutare la diversa gradualità con la quale una azione può essere compiuta, e l'introduzione di opportune scale di valori operata da Schank non risulta in grado di adattarsi alla realtà descritta.

Altre critiche riguardano la struttura stessa dei vari *ROLE* e la consistenza dei vari assiomi. Ad esempio l'*ACT MOVE* che indica il movimento di un corpo non è inserito nell'*ACT PTRANS* come dal punto di vista teorico dovrebbe essere.

Tale modello si propone come una sorta di interlingua non prendendo però adeguatamente in considerazione come la realtà stessa sia modellata attraverso la lingua presa come riferimento.⁵³

Esamineremo ora un altro modello esemplificativo dei vari tentativi producibili per rappresentare una realtà e le informazioni in essa contenute.

Teoria dei Frame

Essenzialmente un frame è una struttura rappresentativa di una determinata conoscenza, che definisce e descrive una classe di entità e situazioni in base a determinati criteri predefiniti. Un tipo di approccio potrebbe essere (a titolo esclusivamente esemplificativo) il seguente:

La frase:

Il 3 di agosto 2013

può essere rappresentata mediante un frame con le seguenti caratteristiche :

DATA	
Anno	[yyyy]
Mese	[m-name ∈ {gennaio, ... , dicembre}]
Mese-numero	[m-number = integer con 0 < m-number <= 12; procedura <i>pl</i> per calcolare m-number, dato m-name]
Giorno	[d-name ∈ {lunedì, ... , domenica}]
Giorno-numero	[d-number = integer 0 < d-number <= length(Mese)]

Tab. 4 - Esempio di un frame non ancora assegnato

Si individuano i vari slot e le informazioni presenti: il frame riempito a partire dai dati a disposizione, produce i seguenti risultati:

DATA	
Anno	2013
Mese	Agosto
Mese-numero	8
Giorno	
Giorno-numero	3

Tab. 5 - Frame *istanziato*

⁵³ Si faccia riferimento all'ipotesi Sapir-Whorf: la categorizzazione linguistica non è solo frutto del nostro modo di organizzare l'esperienza, ma ne è, al contempo, la discriminante: chi, infatti, "conosce" linguisticamente il mondo in un certo modo ne sarà influenzato di conseguenza, ovvero sia il modo di esprimersi influenza il modo di pensare.

Secondo Marvin Minsky⁵⁴, l'ideatore principale di tale teoria, i frame potrebbero essere strutturati e ordinati in un vero e proprio sistema che a sua volta entrerebbe a far parte di una rete di sistemi gerarchici, basati su relazioni di similarità.

Un esempio concreto di tale forma di approccio all'Information Extraction è dato dal sistema FASTUS⁵⁵ sviluppato negli Stati Uniti a opera della SRI International.⁵⁶ È un sistema costruito su cinque livelli di elaborazione: nel primo vengono identificati i nomi propri, le espressioni temporali, i nomi di luogo e i nomi composti. Una particolare analisi viene fatta nei confronti delle parole sconosciute che con molta probabilità possono risultare degli ulteriori nomi propri.

Un ulteriore stadio di elaborazione identifica le frasi nominali, i verbi frasali ed altri elementi come le frasi congiuntive e preposizionali

Per esempio la frase:

Bridgestone Sports Co. said Friday it has set up a joint venture in Taiwan with a local concern and a Japanese trading house to produce golf clubs to be shipped to Japan.

può essere riscritta in un frame del seguente tipo:

Nome Compagnia	Bridgestone Sports Co.
Verbo	Said
Nome	Friday
Nome	It
Verbo	had set up
Nome	a joint venture
Preposizione	In
Luogo	Taiwan
Preposizione	With
Nome	a location concern
And	And
Nome	a japanese trading house
Verbo	to produce
Nome	golf clubs
Verbo	to be shipped
Preposizione	To
Luogo	Japan

Tab. 6 - Secondo stadio del sistema FAUSTUS

Un ulteriore livello di elaborazione produce altri *frame* simili al seguente dove al centro dell'attenzione sono le relazioni tra le varie entità e l'attività svolte da quest'ultime.

⁵⁴ Marvin Lee Minsky (n. 1927) è un informatico e scienziato statunitense specializzato nel campo dell'intelligenza artificiale (AI). È inoltre cofondatore del laboratorio AI presso il MIT e autore di numerosi testi riguardanti l'AI e la filosofia. Minsky è stato consulente per il film *2001: Odissea nello spazio* e viene citato nel film e nel libro tratto dal film.

⁵⁵ FASTUS è un acronimo, leggermente permutato di Finite State Automata-based Text Understanding System. È un sistema per l'estrazione di informazione dai testi correntemente disponibile nella versione inglese e giapponese e sviluppato in Common Lisp. Si consulti il sito: <http://www.ai.sri.com/natural-language/projects/fastus.html>

⁵⁶ L'SRI International fu fondato all'Università di Stanford nel 1946 come un centro di studi per l'innovazione e lo sviluppo economico. È attualmente uno dei più importanti centri di ricerca al mondo.

Tipo di relazione	set up a joint venture
Entità coinvolte	Bridgestone Sports Co. a local concern a Japanese trading house
Attività	...
Quantità	...

Attività svolta	PRODUCTION
Compagnia	
Prodotto	golf clubs
Data di partenza	

Tab. 7 - Quarto stadio del sistema di Information Extraction Faustus

In tale sistema si effettua un lavoro di riduzione di ogni frase alla sua forma attiva analogamente alle procedure tipiche della Grammatica Generativo Trasformativa.

Per esempio tutte le frasi del seguente tipo:

Cars are manufactured by GM.
Cars are to be manufactured by GM.
GM., which manufactures cars

sono trasposte nella seguente semplice forma attiva:

GM manufactures cars.

Tale forma è rappresentabile in un frame analogo a quello precedentemente descritto.

È in questo caso estremamente significativo che per gestire tali frasi sono necessari dei moduli di più basso livello in grado di operare a livello morfologico prima, e poi sintattico.

Espressioni come “X are manufactured by Y”, a questi livelli, devono essere automaticamente trasformate in “Y manufactures X”, con una gestione appunto morfologica del verbo presente nella frase.

Valutazione.

La valutazione delle prestazioni di IR e IE è effettuata in termini di Recall, Precision e F-Measure, nel confronto con un testo annotato manualmente. Nell'Information Retrieval la *precisione* si definisce come il rapporto tra i documenti rilevanti reperiti, e il totale dei documenti reperiti, il *recall* è invece il rapporto tra i documenti rilevanti reperiti e il totale dei documenti rilevanti a disposizione, siano essi reperiti dal sistema oppure no. In termini pratici, avere un alto parametro di *precisione* implica per l'utente non dover sfogliare tra molti risultati inutili per arrivare a quelli corretti, mentre un parametro elevato di *recall* implica che se l'informazione ricercata esiste nella base di conoscenza accessibile, allora viene reperita con elevata probabilità. Questi valori possono essere calcolati analogamente nell'ambito dell'IE considerando i tag assegnati e quelli effettivamente corretti. Ai valori considerati si aggiunge un terzo parametro, come abbiamo menzionato l'F-MEASURE calcolato a partire dai valori di RECALL e PRECISION.

In sintesi:

Recall= numero di tag corretti / numero totale di tag presenti nel testo

Precision= numero di tag corretti nella risposta del sistema/ numero totale di tag presenti nella risposta del sistema

$F = 2 / (1/Recall + 1/Precision)$

5. Teoria del discorso e applicazioni

Considerando il linguaggio come un fenomeno che non solo opera a livello di parole, sintagmi o intere frasi, ma a un livello superiore, dove i singoli elementi costituiscono un insieme correlato e una sorta di struttura, è opportuno descrivere alcuni aspetti della realtà *testuale* secondo le teorie riguardanti la cosiddetta definizione di discorso.

Descriveremo ora i più importanti concetti riguardanti tali teorie e i metodi utilizzati per affrontare questa problematica.

Va anticipato, per altro, che nell'ambito dell'elaborazione di applicazioni per gestire l'interazione uomo-computer (o HCI: Human-Computer Interaction) la costruzione di testi che abbiano carattere di discorso risulta avere delle prerogative a se stanti basate sugli assunti che andremo a descrivere.

Innanzitutto, quindi, alcune nozioni e definizioni terminologiche fondamentali: secondo la definizione riportata dal Longman⁵⁷ un discorso risulta:

- 1) un brano di scrittura a carattere serio riguardante un determinato argomento,
- 2) una seria conversazione o una discussione tra persone,
- 3) il linguaggio utilizzato in un particolare tipo di argomentazione orale o scrittura.



Fig. 20 - Definizioni del termine *discourse* fornita online dal dizionario Longman, Dictionary of Contemporary English

Ciò che Longman intende presumibilmente con il termine “serio” (ma che non dice esplicitamente), è che il testo prodotto non è una collezione casuale di simboli o parole, ma un insieme di frasi correlate, con un determinato significato e che posseggono uno specifico scopo comunicativo.

Abbiamo parlato di frasi correlate aventi un particolare significato poiché un discorso, necessariamente, secondo il senso comune e le teorie più accreditate deve possedere al tempo stesso coesione e deve essere coerente.

Un discorso normalmente manifesta la propria coesione nella maniera con la quale le sue unità testuali sono collegate insieme. La coesione si verifica dove l'interpretazione di alcuni elementi del discorso dipende da quella di altri, implicando l'uso di forme linguistiche

⁵⁷ Il dizionario è disponibile online all'indirizzo <http://www.ldoceonline.com/dictionary>

alternative (ad esempio le abbreviazioni) tali da venir riconosciute e comprese dall'ascoltatore o dal lettore, e riferibili a elementi precedentemente menzionati nel testo scritto parlato.

Consideriamo la seguente frase dal romanzo *Orgoglio e pregiudizio*, di Jane Austen.⁵⁸

Elizabeth looked archly, and turned away. Her resistance had not injured her with the gentleman.

Anche se nel testo non è espresso esplicitamente, è ovvio che la seconda frase, quella a partire dal punto, è collegata alla prima, e che il pronome relativo “her” si riferisce a *Elisabeth*. È proprio questo riferimento che assicura la coesione tra le due frasi. Se il testo venisse cambiato sostituendo “her” con “his” nella seconda frase o se la intera seconda frase fosse sostituita dalla frase seguente: “*This chapter is about discourse processing*”, non si verificherebbe nessuna coesione: l'interpretazione della seconda frase in entrambi i casi non dipenderebbe dalla prima frase. Nell'esempio presentato è proprio l'uso del pronome “her” (come abbiamo discusso a proposito dell'Information Extraction e del problema della risoluzione delle anafore) che determina la coesione, anche se una coesione lessicale è ugualmente possibile attraverso una semplice ripetizione di parole, sinonimi, o iperonimi.

Mentre la coesione nei testi risulta molto più di una semplice connessione tra le varie frasi, l'unità testuale che si verifica attraverso l'utilizzo di anafore e ripetizioni lessicali, può determinarsi anche a livello di significato, manifestando la cosiddetta coerenza. Più specificatamente la coerenza si riferisce all'unità di significato tra due unità di discorso che si compongono per produrre il significato globale di un discorso specifico.

(a) George passed his exam. He scored the highest possible mark.

(b) George passed the exam. He enjoyed red wine.

Mentre nella frase (a) troviamo un perfetto modello di coerenza poiché la seconda frase è una sorta di elaborazione della prima affermazione, il testo (b) risulta incoerente e quantomeno privo di un significato globale. Tralasciando le forzate interpretazioni ipotetiche di quest'ultimo testo, è possibile riscontrare la notevole diversità nella correlazione delle frasi dei due esempi.

Un discorso può prendere la forma di un monologo quando lo scrittore o il parlante del testo in esame è anche l'autore di tale testo. Una particolare forma di discorso è il dialogo, del quale ci occuperemo in modo esteso in questa ricerca e dove esiste interazione e *conversazione*, tra due o più partecipanti. Un'altra forma di discorso è il *multiparty discourse* che in genere si verifica tra più di due individui nelle conferenze o nei meeting.

Organizzazione del testo.

A illustrare come uno specifico discorso può essere organizzato si consideri la seguente voce del *Hamlyn Pocket Dictionary of Wines* per la definizione di un vino: il Flagey-Echezeaux

Flagey-Echezeaux (*France*). Important red wine township in the Cote de Nuits with two front-ranking vineyards, Echezeaux and Grands Echezeaux. The first produces a fine rich, round wine and the second, which is not a single vineyard but a group, is also capable of producing fine wine but, like other divided properties, the quality of its wine is variable.

The lesser wines of Flagey-Echezeaux are entitled to the appellation Vosne-Romanee.⁵⁹

⁵⁸ Austen Jane, *Pride and Prejudice*, T. Egerton (1813)

L'autore non organizza o ordina le frasi nel testo in maniera casuale, ma piuttosto pianifica una struttura generale in cui le varie frasi sono inserite o prodotte. Per l'esempio considerato, l'impostazione scelta è quella tipica delle definizioni di vocabolario. L'autore identifica il Flagey-Echezeaux descrivendone la provenienza ("important red wine township in the Cote de Nuits"), e quindi menzionandone due dei costituenti ("Echezeaux and Grands Echezeaux"), denota entrambi, e procede oltre nelle varie specificazioni.

La presenza di uno schema per impostare il discorso è un elemento ampiamente sfruttato nell'ambito della gestione computerizzata dei testi e soprattutto nell'ambito della cosiddetta NLG (Natural Language Generation). Secondo Kathleen McKeown è possibile parlare di un vero e proprio *schema*⁶⁰ nel quale possiamo distinguere una parte attributiva, una identificativa, una costitutiva, a cui si aggiunge una parte contrastiva.

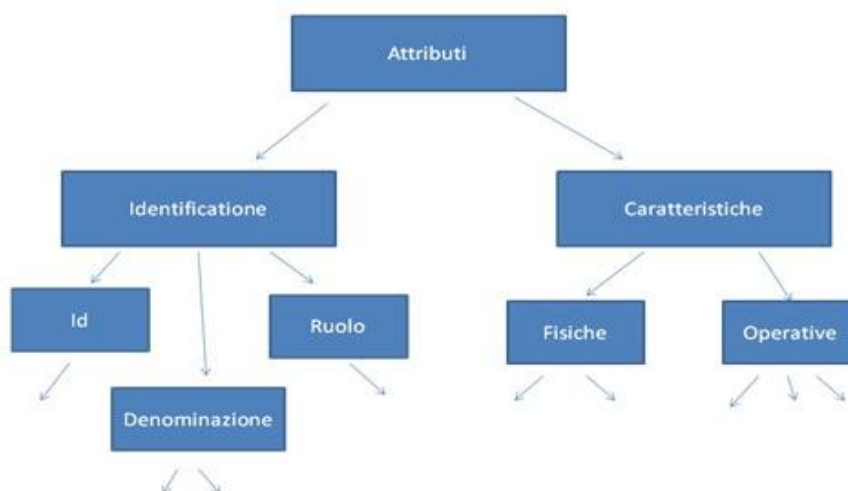


Fig. 21 - Aspetti dello *Schema* proposto dalla McKeown per definire la gerarchia degli argomenti

Ogni schema consiste di una sequenza di predicati che lo identificano attraverso una possibile sequenza: si passa da un predicato identificativo, a quello attributivo, e a quello costitutivo.

Ad esempio lo schema identificativo individua una strategia utilizzata per produrre definizioni: l'elemento oggetto di studio viene presentato come membro di una classe generica, ne vengono elencati i suoi attributi, vengono proposte delle analogie esplicative e degli esempi chiarificatori. La seguente frase, sempre relativa a un vino è esemplificativa:

Eltville (Germany)

(1) An important wine village of the Rheingau region. (2) The vineyards make wines that are emphatically of the Rheingau style, (3) with a considerable weight for a white wine. (4) Taubenberg, Sonnenberg and Langenstuck are among vineyards of note.

Classificazione dell'esempio:

⁵⁹ Paterson John, *The Hamlyn Pocket Dictionary of Wines*. Hamlyn (1980)

⁶⁰ McKeown parla di *schemata*.

- (1) Identificazione di appartenenza a una classe
- (2) Attributi
- (3) Analogie e comparazioni
- (4) Esempi esplicativi

È riscontrabile empiricamente che i testi scientifici e i loro abstract posseggono una struttura predefinita che può essere sfruttata dai programmi applicativi che si occupano di NLP. Gli schemi individuano dei pattern nella struttura dei testi, che sono frequentemente usati da chi scrive e che non riflettono le variazioni prodotte dallo stile di scrittura individuale. Descrivono le norme seguite per raggiungere lo scopo del discorso. Secondo Joseph Grimes che parla appunto di predicati retorici, tali predicati possono essere usati ricorsivamente a ogni livello: una combinazione di predicati può essere sfruttata per realizzare ad esempio una sola delle funzioni descritte.

Un altro esempio di come i testi siano organizzati viene dato dai documenti informativi a carattere medico (le prognosi riguardanti determinate malattie e gli stessi bugiardi); gli argomenti sono definiti secondo le tipiche strutture del discorso: non sono presentati in maniera casuale o arbitraria ma seguono un determinato ordine. Possiamo anche in questo caso parlare di predicati retorici. I testi informativi, definiscono inizialmente delle informazioni di background riguardanti la medicina in oggetto, quindi presentano istruzioni riguardanti il suo dosaggio, seguite da una spiegazione incentrata sui benefici che tale medicina riesce ad apportare. Questi testi elencano infine delle informazioni riguardanti gli effetti collaterali della stessa medicina, e quindi procedono con ulteriori suggerimenti e istruzioni di avvertimento riguardanti le varie circostanze in cui è preferibile avvertire un dottore specializzato. Di recente sono inserite anche informazioni riguardanti i siti on-line di riferimento.

Algoritmo di segmentazione del testo

Gli esempi appena presentati illustrano come un discorso può essere organizzato e segmentato sulla base di differenti argomenti. Per molte pratiche realizzazioni di programmi per l’NLP la segmentazione automatica di un testo in base ai suoi argomenti trattati, è cruciale. Tra i vari algoritmi che sono stati sviluppati per risolvere questo problema l’approccio *TextTiling*⁶¹ risulta il più conosciuto: tale algoritmo non cerca di definire alcuna relazione gerarchica o altro tipo di dipendenza che sussiste tra i vari segmenti o argomenti del testo (unità di discorso) secondo la teoria delle strutture retoriche. Segmenta, invece, i vari testi in una sequenza di paragrafi, ciascuno dei quali è focalizzato verso un ben distinto scopo al momento del discorso in questione. Lo scopo dell’algoritmo è cioè quello di suddividere il testo secondo partizioni contigue (il *tiling* delle finestre dei diversi sistemi operativi), che non si sovrappongono e che insieme costituiscono per intero il testo.

Secondo tale approccio, la struttura di un documento è trattata come una sequenza di discussioni che si verificano nel contesto di uno o più argomenti principali. Quest’esempio è illustrato da Marti Hearst attraverso l’esame di un articolo scientifico dal titolo “Stargazers”

⁶¹ Il *TextTiling* è una tecnica per suddividere automaticamente dei testi in unità formate da più paragrafi in corrispondenza a determinati argomenti. L’approccio utilizza una analisi quantitativa dei vari testi. Hearst Marti, “TextTiling: Segmenting Text into Multi-Paragraph Subtopic Passages”, *Computational Linguistics*, 23 (1), pp. 33-64 (1997).

il cui il principale argomento è l'esistenza della vita sulla terra. I suoi contenuti possono essere descritti secondo la definizione dei seguenti nove sotto-argomenti:

- 1–3 Intro – the search for life in space
- 4–5 The Moon's chemical composition
- 6–8 How early Earth–Moon proximity shaped the Moon
- 9–12 How the Moon helped life evolve on Earth
- 13 Improbability of the Earth–Moon system
- 14–16 Binary/trinary star systems make life unlikely
- 17–18 The low probability of non-binary/trinary systems
- 19–20 Properties of Earth's Sun that facilitate life
- 21 Summary

Nell'esempio la numerazione fa riferimento ai vari paragrafi; l'approccio ipotizza che un particolare insieme di elementi lessicali viene utilizzato all'interno della discussione riguardante un determinato argomento, un *subtopic*, e quando tale subtopic muta, si verifica un significativo cambiamento nel vocabolario stesso, nella proporzione statistica in cui i vari termini vengono utilizzati.

14	form	1	111	1	1				1	1	1	1	1	1	1	1	1	1	1	1		
8	scientist				11		1	1		1		1	1	1	1							
5	space	11	1	1										1								
25	star	1			1					11	22	111112	1	1	1	11	1111			1		
5	binary									11	1		1							1		
4	trinary									1	1		1							1		
8	astronomer	1			1					1	1		1	1	1	1	1					
7	orbit	1				1					12	1	1									
6	pull					2	1	1				1	1									
16	planet	1	1		11		1		1			21	11111						1	1		
7	galaxy	1									1			1	11	1				1		
4	lunar			1	1	1		1														
19	life	1	1	1					1	11	1	11	1	1			1	1	1	111	1	1
27	moon		13	1111	1	1	22	21	21	21		11	1									
3	move									1	1	1	1									
7	continent								2	1	1	2	1									
3	shoreline											12										
6	time				1				1	1	1	1	1									1
3	water								11			1										
6	say						1	1		1		11			1							
3	species								1	1	1											
Sentence:		05	10	15	20	25	30	35	40	45	50	55	60	65	70	75	80	85	90	95		

Fig. 22 - Distribuzione di frequenza di determinati termini nel testo *Stargazers*. I vari numeri in corrispondenza di una colonna (corrispondente a una frase) indicano che in quella frase il termine è presente.

L'algoritmo TextTiling definisce tre principali categorie di elementi lessicali individuarli all'interno di un testo:

- 1) parole che si presentano frequentemente all'interno del testo, e che sono indicative dell'argomento trattato;
- 2) parole che sono meno frequenti ma più uniformi nella distribuzione di frequenza stessa: tali parole non forniscono rilevanti informazioni nei confronti della suddivisione trattata;
- 3) gruppi di parole che sono raggruppabili insieme, con un'altra densità di frequenza in alcune parti del testo e bassa densità e in altre parti. Questi gruppi di parole sono indicative della struttura dei sotto argomenti.

Il problema della segmentazione è quindi il problema di determinare dove questi raggruppamenti di parole iniziano e finiscono. Hearst propone un algoritmo di

comparazione (block comparison) per risolvere questo problema. In tale algoritmo, viene definito un blocco come una sequenza di n frasi contigue. I blocchi vengono comparati due a due, secondo il criterio della loro similarità lessicale e la distanza tra ciascun paio di blocchi viene definita secondo un valore numerico che costituirà il cosiddetto *lexical score*. Tante più parole i due blocchi hanno in comune tanto più alto risulterà il punteggio assegnato alla ipotetica variabile “gap”. Se un gap con un basso punteggio lessicale è seguito e preceduto da gap con elevati punteggi, molto probabilmente, il blocco riferito al primo gap con un basso punteggio indicherà un cambiamento nel vocabolario, che molto probabilmente corrisponderà a un cambiamento del tema all’interno.

Nella loro iniziale rappresentazione, i blocchi sono rappresentati da vettori e il punteggio lessicale è calcolato come il prodotto dei due vettori stessi, normalizzato.⁶² Il sistema TextTiling identifica cambiamenti di argomento in base ovviamente a un livello di soglia cui farà riferimento il punteggio del gap.

Le prestazioni di tale sistema hanno riscosso notevoli consensi anche se esistono metodi alternativi per calcolare il *gap* tra i vari blocchi di testo: lo stesso Hearst ne ha proposti alcuni di tra cui le *lexical chains*, le *vocabulary introductions* e altre ulteriori varianti.

La LSA (Latent Semantic Analysis)⁶³ è una variante del *TextTiling* con delle prestazioni decisamente migliorate. In questo approccio il miglioramento è determinato dal fatto che l’analisi semantica sostituisce ciascuna parola all’interno dei vari blocchi di testo con appropriati gruppi di parole che ricorrono contemporaneamente in essi.

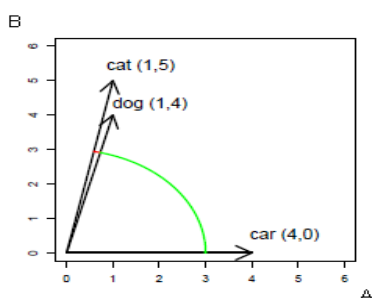
I vettori iniziali sono perciò convertiti in matrici di vettori nelle quali la similarità è calcolata in base alla somma dei coseni tra ciascuno di tali vettori.⁶⁴

Hideki Kozima⁶⁵ descrive un approccio molto simile alla segmentazione del discorso in cui ciascun blocco di testo all’interno di un documento viene associato a un *lexical cohesion profile*, una sorta di record rappresentante la coesione lessicale (similarità semantica) delle

⁶² Supponiamo di considerare un blocco di frasi in base alla presenza in esso di tre termini “cane”, “gatto”, “auto”. Se nella frase “cane” è presente due volte, “gatto” una volta, “auto” zero volte potremo rappresentare il blocco di frasi come il seguente vettore numerico [2 1 0] che differirà da un vettore [0 1 1] indicante un blocco in cui il termine “cane” è presente zero volte, il termine “gatto” una volta, il termine “auto” una volta. Il prodotto dei due vettori è una delle operazioni possibili con queste strutture numeriche.

⁶³ L’LSA è stata brevettata nel 1988 (US Patent 4,839,853) da un team di studiosi tra cui Scott Deerwester, Susan Dumais, George Furnas, Richard Harshman, Thomas Landauer, Karen Lochbaum and Lynn Streever. nel contesto delle applicazioni di Information Retrieval, è talvolta chiamata Latent Semantic Indexing (LSI)

⁶⁴ Supponiamo di avere due gruppi di frasi A e B. Nel gruppo A il termine “cat” ricorre una volta, nel gruppo B ricorre cinque volte. Inoltre nel gruppo A il termine “dog” ricorre una volta mentre nel gruppo B, 4 volte. Il termine “car” ricorre invece 4 volte nel gruppo A e zero nel gruppo B. Uno spazio semantico in base ai due gruppi di frasi (e la similarità semantica degli stessi) può essere descritto in uno schema grafico come il seguente.



⁶⁵ Kozima Hideki, “Text segmentation based on similarity between words”, *Proceedings of the 31st Annual Meeting (Student Session) of the Association for Computational Linguistics* (1993)

parole all'interno del blocco. Livelli bassi di coesione lessicale all'interno di un blocco indicano il verificarsi di un cambiamento dei subtopic.

Coerenza del discorso.

Nel capitolo precedente abbiamo discusso le varie problematiche inerenti la struttura di un discorso: la maniera con la quale le varie unità di un testo sono connesse per formare delle significative relazioni in grado di contribuire all'accettabilità del discorso stesso. Tratteremo ora alcune teorie riguardanti la coerenza testuale che hanno svolto un ruolo influente nell'ambito di diversi modelli computazionali riferiti al discorso, e di diverse applicazioni di elaborazione del linguaggio naturale.

Teoria di Hobbs

Nel modello di Jerry Hobbs,⁶⁶ un discorso comprende nella sua struttura, unità di testo in relazioni di coerenza tra di loro.

Si consideri il seguente esempio dove la seconda frase è una elaborazione della prima e una sorta di continuazione logica di quest'ultima:

(a) Go down Washington Street. Just follow Washington Street three blocks to Adams Street.

Facendo riferimento al lavoro di Grimes⁶⁷ e di Halliday,⁶⁸ Hobbs propone un insieme di 12 relazioni di coerenza tra le quali causa, valutazione, contesto, parallelismo, elaborazione.

Si consideri come ulteriore esempio la seguente frase:

(b) John can open Bill's safe. He knows the combination.

Potremmo rappresentare le due frasi secondo il seguente schema in una terminologia analoga a quella del calcolo dei predicati:

can(John, open(Safe))
know(he, combination(Comb,y))⁶⁹

E dopo alcune semplificazioni è possibile ricostruire la notazione:

know(he, cause(dial(z, Comb, y), open(y)))
He knows that z's dialing the combination of y causes it to open

nella quale gli elementi sono connessi tra loro senza nessuna interruzione interpretativa.

Un altro modello di organizzazione testuale è derivato dalla cosiddetta Rhetorical Structure Theory (RST)⁷⁰ nella quale le relazioni tra due o più parti di testo non

⁶⁶ Hobbs 1979, Jerry R. Hobbs (n. 1942) è un eminente ricercatore nel campo della linguistica computazionale, dell'analisi del discorso e dell'intelligenza artificiale.

⁶⁷ Grimes Joseph, *The Thread of Discourse*. Mouton (1975).

⁶⁸ Halliday M., Hasan R., *Cohesion in English*, Longman, Londra (1976).

⁶⁹ Nel cosiddetto *Calcolo dei predicati del primo ordine* possiamo distinguere fondamentalmente tra variabili (A, B, X, Y...), costanti di funzioni (relazioni del tipo "cuginoDi", "distanzaTra") e altri operatori (sposta(X,Y), sposta X su Y).

sovrapponibili, si suddividono in parti denominate nucleo e satelliti. Il nucleo ha ovviamente un ruolo centrale e indipendente. Il satellite ha un ruolo meno centrale e spesso la sua interpretazione è connessa a quella del nucleo. Mann e Thompson descrivono un insieme di 25 relazioni.

Nella seguente frase individuiamo una relazione di “dimostrazione” o “prova a supporto” poiché la seconda parte della frase (il satellite) è utilizzabile per supportare l’antefatto:

(a) Probably the most extreme case of Visitors Fever I have ever witnessed was a few summers ago.

(b) Probably the most extreme case of Visitors Fever I have ever witnessed was a few summers ago when I visited the relatives in the Midwest.

Nel seguente frammento di articolo tratto da Scientific American del maggio 1986 e intitolato “Darwin as a Geologist” le varie frasi:

1) Darwin as a Geologist 2) He tends to be viewed now as a biologist, 3) but in his five years on the Beagle his main work was geology, 4) and he saw himself as a geologist. 5) His work contributed significantly to the field.

possono essere rappresentate nel seguente diagramma RST:

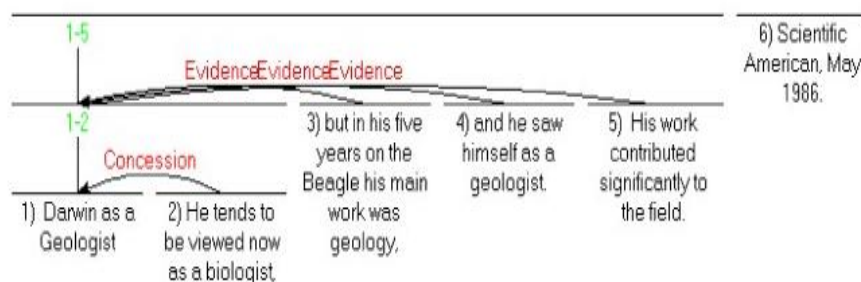


Fig. 23 - Rappresentazione di una frase secondo le teorie la Rethorical Structure Theory

Il nucleo della frase è individuabile nei sintagmi (1-2) e i vari satelliti nei sintagmi successivi.

⁷⁰ Mann William, Thompson Sandra, “Rhetorical structure theory: a theory of text organization.” RS-87-190, *Information Sciences Institute* (1987). Indicazioni a proposito possono essere tratte all’indirizzo: <http://www.sfu.ca/rst/01intro/index.html>

6. Natural Language Generation

La generazione del linguaggio naturale, (Natural Language Generation - NLG) è il procedimento mediante il quale i tratti del nostro pensiero sono resi nella forma di un linguaggio scritto o parlato. A partire da considerazioni filosofiche, neurologiche, psicologiche e naturalmente linguistiche, in questo capitolo, esamineremo ciò che intendiamo per “generazione” da una prospettiva computazionale, e cioè in relazione alle possibili applicazioni di intelligenza artificiale e linguistica computazionale utilizzabili per produrre un determinato testo.

Il “generatore” (l’equivalente di una persona con qualcosa da dire) risulta nel nostro caso un computer, o meglio un programma per computer, il cui compito inizia con l’intenzione di “comunicare” qualcosa, e si sviluppa nel determinare il contenuto di ciò che verrà detto, selezionare la lingua, le parole e l’organizzazione retorica di queste ultime, allo scopo di produrre un testo più o meno esteso.

Potremo avere di tale testo una realizzazione scritta (in una formattazione grafica) o parlata (attraverso una gestione degli elementi prosodici e più generalmente fonici corrispondenti).

Allo stato attuale della tecnologia, ciò che un generatore produce può consistere in una singola parola o frase, data in una risposta a una domanda, o in un’etichetta informativa aggiunta a un diagramma attraverso vari procedimenti di adattamento grafico, o anche nella produzione di risposte a domande più o meno complesse nell’ambito di un dialogo, o infine come spiegazione dettagliata o riassunto di un determinato argomento.

Cenni storici

L’NLG ovviamente costituisce un settore della linguistica computazionale da quando la disciplina è sorta e si sviluppa quindi a partire dagli anni ‘50 come un aspetto marginale del problema della traduzione automatica.

Ciò nonostante, in un certo senso possiamo dire che l’NLG è tra i primi aspetti del *language processing* ad essere gestito implicitamente dai computer.

Il più tradizionale e famoso programma in linguaggio “C”, il programma “Hello World” è un programma di generazione. Produce una frase di saluto in inglese, semplice ma “ineccepibile” dal punto di vista grammaticale.

```
// Hello World.cpp : Definiert den Einsprungpunkt für die Konsolenanwendung.
//

#include "stdafx.h"

int main(int argc, char* argv[])
{
    printf("Esempio di produzione di stringhe!\n\n");
    printf("Hello World!\n\n");
    char* c;
    scanf(c);
    return 0;
}
```

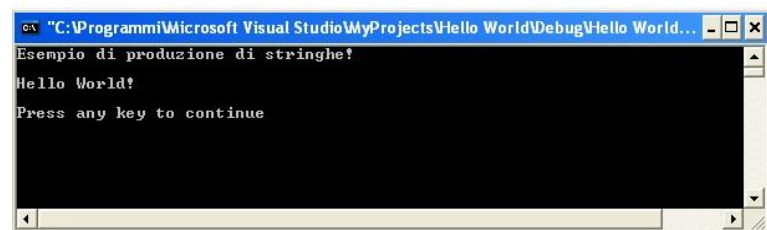


Fig. 24 - Listato del programma Hello World e schermata della sua esecuzione.

Ovviamente il frammento di testo “Hello World” non è prodotto dal computer stesso ma in maniera diretta dal programmatore che può decidere le eventuali varianti del messaggio di saluto del programma.

Questo approccio, che considereremo più in dettaglio in seguito, è definito come *canned text generation*: è di facile realizzazione ma difficile da adattare a nuove situazioni senza l'intervento di un programmatore.

Ciò nonostante, già negli anni '60, furono sviluppati generatori più complessi di frasi casuali, spesso utilizzati a scopo di verifica grammaticale, e, poco più tardi, si cercarono di realizzare dei veri e propri parlanti artificiali. All'interno di alcune primordiali interfacce dialoganti si cercò cioè di comporre sistemi in grado di dare risposte relative a domande effettuate in un ristretto contesto, ad esempio un limitato database di informazioni. In questo periodo i vari generatori venivano già sviluppati con modelli di grammatica provvisti di una notevole sofisticatezza: i testi che venivano realizzati, sebbene di ridotta estensione, risultavano di notevole qualità.

Si segnalano a proposito i contributi di Neil Goldman⁷¹ (nella cui tesi di Ph.D. realizzava un sistema per esprimere con parole appropriate dei concetti astratti) e di Anthony Davey⁷² in grado di realizzare una descrizione linguistica di partite di tic-tac-toe (il nostro filetto).

Negli anni '80, le ricerche nell'ambito della realizzazione di testi ad opera di computer divennero più consistenti e la NLG si trasformò in una disciplina a se stante, con scopi e prospettive specifiche.

Comparvero i lavori di Kathleen McKeown e di Douglas Appelt⁷³ e gli studiosi orientarono i propri sforzi nell'analisi di problematiche in contesti limitati; nel 1983 si tenne poi il primo International Workshop sulla Natural Language Generation.

Negli anni '90, tra i tanti sviluppi della disciplina, segnaliamo il sistema FoG (Forecast Generator)⁷⁴ il primo sistema utilizzato praticamente e che forniva una generazione di previsioni del tempo in due lingue. La “prima previsione” fu effettuata il 27 febbraio 1992.

Oggi il trend di interesse verso l'NLG è in continua crescita a causa delle medesime ragioni per cui, lo stesso interesse per la Linguistica Computazionale, in ogni suo aspetto, sta crescendo. Di particolare rilievo risulta inoltre il tentativo di combinare la generazione di testi con la generazione di immagini e grafici allo scopo di realizzare sistemi in grado di contenere, elementi grafici e testuali congiuntamente.

La generazione di un testo in rapporto alla sua comprensione.

Il confronto di queste due attività risulta molto utile poiché anche a distanza di circa 40 anni, dai primi tentativi rivolti a realizzare un sistema di *generazione*, l'NLG viene ritenuta una semplice variante della *comprensione*, mentre invece dovrebbe essere considerata una problematica diametralmente opposta, incentrata sulla costruzione e sulla pianificazione dei contenuti da esprimere.

Nella comprensione ci si muove, cioè, in un certo senso, dal testo alle intenzioni con le quali il testo è stato costruito. Il primo processo di elaborazione risulta una scansione sequenziale delle parole, durante la quale la forma del testo viene individuata e modellata.

⁷¹ Goldman Neil M., *Conceptual Generation*, in Schank Roger C., Riesbeck Christopher K., *Conceptual Information Processing*. American Elsevier, New York (1975).

⁷² Davey Anthony, *Discourse Production*, Edinburgh University Press, Edinburgh (1979).

⁷³ Appelt Douglas, *Planning English Referring Expressions*, Cambridge University Press, New York (1985).

⁷⁴ Il sistema FoG fu sviluppato dall'CoGenText, una software house specializzata nella Natural Language Generation per la Environment Canada (un servizio meteo canadese)

Le maggiori problematiche sono causate ovviamente dall'ambiguità del linguaggio (una *forma* può far riferimento a una vasta gamma di significati alternativi), e da una sorta di limitata specifica implicita in qualsiasi espressione linguistica: nella lettura di una quantità ridotta di segni e di informazioni grafiche oggettivamente presenti, il referente deve essere in grado di individuare tutta una gamma di informazioni ulteriori e complementari in base alle caratteristiche del contesto dove l'espressione si è realizzata.

Nella NLG si verifica invece un flusso di informazioni di tipo opposto: procediamo cioè dalle intenzioni di un ipotetico parlante, al testo, al contenuto, alla forma di quanto si vuole esprimere.

La generazione di un testo, si attua quindi nella produzione di scelte tra le varie alternative presenti all'interno del linguaggio (ad esempio la scelta delle parole e della struttura sintattica con cui collegarle), e quindi infine nella deliberata costruzione di un prodotto con la sua prestabilita forma grafica (in base ai caratteri dell'alfabeto di riferimento): il problema dell'ambiguità nella generazione non si verifica comunque nelle forme tipiche della comprensione.⁷⁵

In definitiva, cioè, il problema di un sistema di generazione del linguaggio naturale è scegliere come evidenziare le proprie inferenze, tra una gamma di possibilità ipoteticamente infinita: scegliere cioè quali informazioni introdurre e quali omettere.

Da questo presupposto è ragionevole ipotizzare che, come abbiamo in parte definito, il processo di generazione può essere organizzato in base a stadi di lavoro, disposti in ordine inverso al processo di comprensione.

Parleremo perciò di pragmatica (individuazione dello scopo per il quale vogliamo generare un determinato testo), di semantica (in un certo senso il significato di ciò che vogliamo comunicare), passeremo quindi a definire i metodi per organizzare le strutture linguistiche utilizzabili nel discorso (sintassi), e infine ai sistemi per la produzione del testo finale (realizzazione morfologica o fonica). Un ulteriore stadio del processo di generazione (nel caso di una generazione in forma scritta) può convertire i paragrafi e le frasi astratte del documento prodotto nella fase precedente in una particolare struttura per la gestione e la diffusione del documento stesso. Ad esempio per la trasformazione di tale documento in codice HTML si potrebbero inserire annotazioni di markup, (<p> per indicare un paragrafo, <h1> un particolare formato del carattere) onde permetterne una corretta lettura da parte di un eventuale browser.

*Un approccio ludico. Douglas R. Hofstadter e la Natural Language Generation*⁷⁶

Ispirandosi ad alcuni esempi dei cosiddetti "Haiku a Calcolatore", programmi per la scrittura di Haiku giapponesi, Hofstadter si cimentò nella elaborazione di un suo programma di generazione: partiva da una grammatica in cui la struttura generale della frase era scelta a caso, per poi produrre parole composte con meccanismi più o meno aleatori. Il programma era simile alla proverbiale scimmia seduta davanti a una macchina da scrivere, che però operava simultaneamente a vari livelli della struttura linguistica e non solo al livello delle lettere. L'utilizzo di un lessico deliberatamente umoristico, produsse una quantità di enunciati privi di senso, alcuni dei quali avevano una struttura molto complicata, mentre altri erano abbastanza brevi. L'effetto era decisamente surreale e a tratti ricordava proprio la

⁷⁵ È necessario per altro notare che ambiguità di ogni tipo sono comunque inavvertitamente introdotte all'interno di ogni messaggio nonostante le intenzioni di chi lo produce.

⁷⁶ Hofstadter Douglas R., *Gödel, Escher, Bach: un'Eterna Ghirlanda Brillante*, Basic Books (1979)

poesia haiku. Al sistema creato vennero aggiunte poi parole italiane con alcune forme elementari di grammatica italiana. Alcune delle frasi risultarono le seguenti:

Vogliamo suonare un pranzo così male. Noi non possiamo toccare il silenzio. Non possiamo camminare dolcemente, ma possiamo ridere e poi dobbiamo starnutire.

Una grande fanciulla maschile mangia il brutto brufolo peloso. Molti battono i pianoforti quando le voci sorridono, ma loro mentiscono male.

Io posso amare una ragazza schifosa. Io mentisco.

Le altre fanciulle maschili possono vender-si bene.

Le altre sabbie sbadate possono cercar-si fantasticamente.

Tu dici che puoi dormire meglio senza te. Tu soffri facilmente con te e adesso cantiamo bene su noi. Dobbiamo gridare.

Ridete. Voi dite che non pensate che voi potete lasciare un uccello moltissimo a voi. Voi vendete il nostro topo. Volete amrvi più tardi. Voi rompete una luna.

La cosa morta non deve sorridere bene.

Malgrado il notevole divertimento nell'impatto iniziale con tali generazione si perde interesse con queste ultime proprio perché incapaci di continuare a sorprendere, una volta individuato il meccanismo aleatorio.

Un primo esempio "funzionale": ERMA

Il programma *ERMA*⁷⁷ sviluppato nei primi anni '70 del novecento simulava analogamente a Weizenbaum le produzioni testuali del paziente di un eventuale psicanalista.

Il lavoro fu effettuato in collaborazione tra John Clippinger e Richard Brown e pubblicato dagli studiosi nelle rispettive tesi di Phd.

Un esempio di testo prodotto da Erma è il seguente:

You know for some reason I just thought about the bill and payment again. (*You shouldn't give me a bill.*) <Uh> I was thinking that I (*shouldn't be given a bill*) of asking you whether it wouldn't be all right for you not to give me a bill. That is, I usually by (*the end of the month know the amount of the bill*), well, I immediately thought of the objections to this, but my idea was that I would simply count up the number of hours and give you a check at the end of the month⁷⁸

Il paragrafo prodotto è il risultato di un complesso modello computazionale in grado di riprodurre i processi intellettivi del paziente. Il programma aveva una capacità di multiprocessing: l'agente generante poteva continuare a pensare e pianificare un discorso,

⁷⁷ ERMA fu realizzata da John Clippinger dal 1973 al 1977. Il nome è tratto dal suono delle interiezioni effettuate da un ipotetico parlante durante la conversazione (err-umm-ah). Il programma venne scritto nel linguaggio CONNIVER.

⁷⁸ Il testo è generato a partire da un modello di conversazione dove effettivamente un paziente conversa con il proprio terapeuta (con caratteristiche analoghe al testo presentato): "You know, for some reason, I huh, just thought about, uh the bill and payment again. That (pause 2 seconds) I was, uh, tghinking that I -- of, asking ...", In Clippinger John, *Speaking with Many Tongues: some Problems in Modeling Speakers of Actual Discourse*, Teleos, Cambridge (1977).

durante l'espressione orale di un'altra porzione di discorso; poteva in un certo senso pensare mentre parlava.

I due studiosi svilupparono un modello di fenomeni di interruzione e ripristino del flusso generato, includendo espressioni come ““uh”” o altre come ““you know””.

Il segmento di testo presentato evidenzia, tra parentesi, ciò che Erma stava pianificando di dire prima che fosse interrotta (una sorta di restart voluto dall'utente) e indotta, in un certo senso a parlare d'altro.

Il sistema era costruito su cinque stadi di formazione del discorso, *Calvino*, *Machiavelli*, *Cicerone*, *Freud* e *Realization*. Ciascuno di questi stadi aveva i propri algoritmi e i propri dati e “esprimevano” delle “opinioni” su ciò che doveva essere detto o comunque generato. Ad esempio lo stadio denominato *Machiavelli* era deputato a ridurre l'impatto delle frasi ritenute più “estreme” nei confronti del ricevente.

La complessità e la particolare ricchezza di un simile programma e del testo presentato evidenzia come, già negli anni '70, il livello di qualità della *generazione* risultava decisamente notevole.

D'altro canto però Erma produsse paradossalmente soltanto quell'esemplare e alcune altre varianti e il lavoro dei due studiosi, nonostante la ragguardevole ricchezza di intuizioni e concrete realizzazioni, non venne in sostanza, più seguito da altri ricercatori.

Ciò nonostante, come già accennato, a partire dagli anni '70, la generazione di testi venne riconosciuta come un campo di studi di interesse fondamentale, e non l'oggetto del contributo anticipatore di alcuni singoli individui. Grazie alle nuove risorse informatiche, che permettevano agli studiosi di lavorare insieme e definire delle mete e delle aspettative comuni, si ebbero importanti e progressivi sviluppi: se i testi che i vari sistemi erano in grado di produrre non risultavano di grande qualità, lo sviluppo della tecnologia, e l'interesse sempre più concreto su tale problematica permise un continuo miglioramento dei risultati ottenuti.

Si strutturò il lavoro del “generatore” in distinti moduli e livelli di rappresentazione, specialmente in riferimento alle varie operazioni: la scelta lessicale, la scelta dell'ordine delle parole e così via. Venne accuratamente studiato il funzionamento dei pronomi e di altre forme analoghe di referenze e si riuscì a sviluppare delle tecniche per aggregare proposizioni minimali a forme sintatticamente più complesse.

“Evoluzione” nell'NLG: *Spokesman*

Diamo ora un esempio di un testo prodotto alla fine gli anni '80 tratto dal sistema *Spokesman* di Marie Meteer⁷⁹. Lo scritto, costruito in un contesto linguistico legato all'ambiente militare, mostra un estratto dell'elaborazione di un ordine operativo (OPORD) attraverso l'ausilio di semplici elementi di formattazione:

1. MISSION

10th Corps defend in assigned sector to defeat the 8th Combined Arms Army.

2. EXECUTION

a. 52d Mechanized Division

Conduct covering force operations along avenues B and C to defeat the lead regiments of the first tactical echelon in the CFA in assigned sector.

⁷⁹ Meteer Marie, *Portable Natural Language Generation using SPOKESMAN*, Rensselaer Polytechnic Institute, New York (1992).

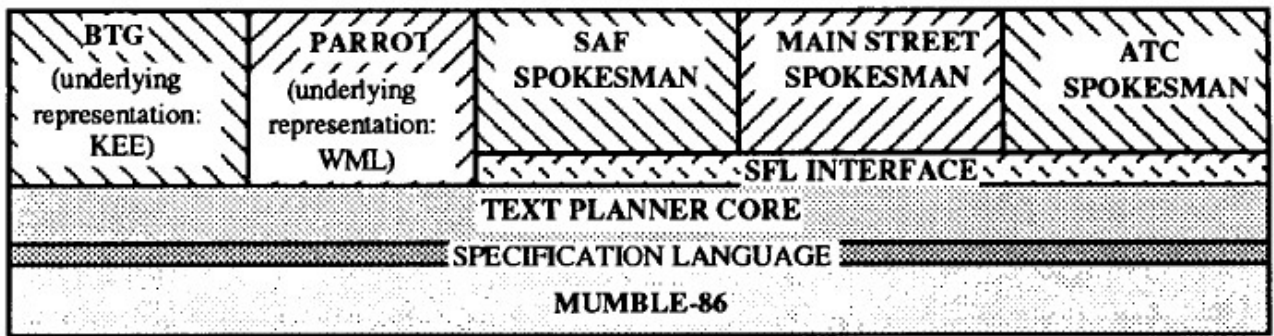


Fig. 25 - Architettura del sistema di generazione Spokesman del 1992. Si noti la presenza dell'ultimo stadio Mumble-86 utilizzato per rendere gli stadi precedenti "trasportabili" da una lingua all'altra (o da un contesto all'altro), secondo le peculiarità della definizione di Mumble-86 appunto.

STREAK

STREAK di Jacques Robin⁸⁰ opera soltanto all'interno di un dominio specifico, in questo caso il linguaggio dello sport, ed è in grado di scrivere dei brevi riassunti di partite di pallacanestro. Come tutti i resoconti sportivi, questo genere di scrittura è caratterizzata da testi praticamente ricchi e densi di informazioni, testi che possono costituire una sfida notevole anche ai migliori sistemi.

In *STREAK*, possiamo osservare e apprezzare la notevole ricchezza di riferimenti a informazioni "storiche": tali informazioni, basate sulla passata conoscenza del sistema (in riferimento a partite già avvenute) risultano avere notevole rilievo nel modo in cui la produzione dei riassunti è stata impostata. Tecnicamente, *STREAK* è un sistema basato su una serie continua di revisioni. Nella prima fase si realizza una rappresentazione dei singoli fatti verificatisi in una partita e presentati nella loro semplicità: tale rappresentazione risulterà un punto di riferimento per i successivi sviluppi generativi.

La seguente nota, può considerarsi come un punto di partenza del lavoro di generazione del sistema *STREAK*.

Dallas, TX—Charles Barkley scored 42 points Sunday as the Phoenix Suns defeated the Dallas Mavericks 123–97.

Il testo iniziale è poi modificato non appena importanti informazioni storiche e altri tipi di informazioni riguardo la partita e le statistiche dei vari giocatori nominati vengono prese in considerazione. La seguente può considerarsi una forma finale della frase precedentemente considerata:

Dallas, TX—Charles Barkley tied a season high with 42 points and Danny Ainge came off the bench to add 21 Sunday as the Phoenix Suns handed the Dallas Mavericks their league worst 13th straight home defeat 123–97.

Possiamo notare che cosa è accaduto: le forme iniziali sono state progressivamente sostituite da frasi che aggiungono informazioni: "scored N points" è diventato "tied a season high with N points".

⁸⁰ Jacques Robin, McKeown Kathleen R., "Empirically designing and evaluating a new revision-based model for summary generation", *Artificial Intelligence* 85, pp. 135-79. Il sistema *STREAK* fu realizzato dal 1993 al 1996. Per ulteriori specifiche si consulti il sito: <http://www.nlg-wiki.org/systems/STREAK>

Inoltre semplici formulazioni sintattiche sono state cambiate e rese più articolate: “*defeat X*” è diventato “*hand X(a) defeat*”;

Data la complessità dei sistemi di generazione di testi disponibili attualmente,⁸¹ la produzione di informazioni dettagliate, anche se semplici, è divenuta un’attività sostanzialmente praticabile con un periodo di programmazione e apprendimento, relativamente breve. Ciò che rende questi moderni sistemi efficaci è comunque la limitazione del dominio in cui essi sono applicabili, dominio dove il corpus di testi analizzati può considerarsi appartenente a una sorta di “sottolinguaggio”.

Struttura di un sistema NLG.

Veniamo ora ad esaminare più in dettaglio e nello specifico le caratteristiche principali della struttura dei vari sistemi disponibili attualmente e a descrivere alcuni dei criteri principali con i quali essi sono effettivamente costruiti.

Come abbiamo ampiamente sottolineato, molto probabilmente, la più importante caratteristica dei sistemi di *Natural Language Generation* consiste nel ruolo centrale svolto dalla necessità di effettuare scelte produttive: scelte che vanno da un livello cosiddetto “alto”, riguardante il contenuto delle informazioni da conferire, ad altre scelte di più basso livello, riguardanti ad esempio l’uso dei pronomi o di altre forme sintattiche e grammaticali. Ma come vengono effettuate tali scelte, e con quali criteri?

Prima di addentrarci in una discussione esaustiva dell’architettura di un sistema NLG, possiamo accennare che i metodi seguiti sono molteplici, spesse volte di immediata applicazione: ad esempio ci si può basare su considerazioni di correttezza linguistica come nel seguente esempio riguardante una scelta, a livello di pronomi:

- 1)
 - a) * Mary read about Mary.
 - b) Mary read about herself.

In questo caso il sistema di generazione riscontra che un pronome riflessivo deve essere usato, qualora l’entità svolgente l’azione è anche quella che la riceve, e in base a regole riguardanti il contesto presentato, genera la seconda frase.

In ogni caso, spesse volte, entrambe le scelte conducono a testi indistintamente corretti. Si consideri infatti:

- 2)
 - a) I bought an apple. I ate it.
 - b) I bought an apple. I ate the apple.

Dal momento che entrambe le scelte conducono a testi analoghi, il sistema di generazione deve decidere fra quest’ultime utilizzando altri criteri. Spesso tali decisioni sono effettuate anche in base a fattori di leggibilità, che a loro volta sono basati su modelli di comprensione del linguaggio, costruiti con criteri propri della psicolinguistica; nell’esempio a cui ci stiamo riferendo alcuni modelli potrebbero suggerire che la prima frase sarà letta in modo più veloce e perciò potrebbe essere quella candidata alla generazione. Ma le decisioni possono anche essere influenzate da altri tipi di criteri; per esempio, l’uso dei pronomi può venire scoraggiato in contesti dove la precisione dei riferimenti è di estrema importanza (in testi

⁸¹ Un elenco è presentato all’indirizzo http://www.nlg-wiki.org/systems/Table_of_NLG_system_builders

come i manuali operativi e in quei manuali dove la precisione dei termini è una prerogativa essenziale).

I modelli presi in considerazione sono comunque tipicamente basati sull'analisi di vari corpora riferiti ai contesti nei quali i sistemi di generazione sono applicati.

In altri casi le decisioni possono anche essere influenzate dalle capacità linguistiche e dalle preferenze proprie del lettore del testo; per esempio un sistema può scegliere, in via ipotetica, la seconda frase dell'alternativa (2) se il lettore non ha una conoscenza fluente dell'inglese.

Prendiamo in esame altri due esempi concreti: le previsioni del tempo del sistema SumTime, e SkillSum, che genera resoconti in ambito educativo.

SumTime

Come abbiamo visto con FoG, i sistemi di generazione di testi sono spesso volte impiegati proprio nella produzione di previsioni del tempo a partire da dati numerici resi disponibili alle varie agenzie che si occupano di tali servizi; questi sistemi hanno come dati d'ingresso un insieme di valori numerici riferiti a parametri come la temperatura, il tasso di precipitazione, la velocità del vento in diversi periodi della giornata, e in diversi luoghi geografici:

Tab. 8 - Previsione delle condizioni del vento per il 19 settembre 2000. Porzioni di testo, da un corpus scritto da un esperto umano e dal sistema SumTime

Time	Wind Dir.	Wind Speed
06:00	SE	11
09:00	SSE	13
12:00	SSE	14
15:00	SSE	15
18:00	SE	18
21:00	SE	23
00:00	SE	28

Corpus (human) text:

SSE 10-15 INCREASING 15-20 BY EVENING AND 25-30 LATER

SumTime text:

SE 9-14 veering SSE 13-18 by mid-afternoon, then increasing

SE 26-31 by midnight

Questi valori numerici sono prodotti da un computer che segue una simulazione numerica delle condizioni atmosferiche, e sono poi modificati da vari addetti e specialisti in base alle proprie conoscenze delle condizioni meteorologiche locali. Da questi valori numerici, il sistema di generazione produce dei testi riassuntivi (a volte in più di un linguaggio) che possono poi essere letti da eventuali utenti (in genere dopo un ulteriore controllo effettuato manualmente).

La tabella 8 mostra un semplice estratto di uno degli insiemi di dati resi disponibili a SumTime, nel quale sono indicate le condizioni della velocità e della direzione del vento durante l'arco di 24 ore.

Allo scopo di generare un testo di output (come quello presentato nell'esempio) in base ai dati in ingresso, il sistema deve effettuare svariate scelte riguardanti sia il contenuto del documento da produrre, sia la sua struttura; tale attività è chiamata *document planning*.

Nel caso specifico, SumTime ha deciso di comunicare le condizioni del vento all'inizio e alla fine del periodo preso in considerazione, e anche a metà della giornata, più precisamente alle ore 15 (“*by mid-afternoon*”).

L'addetto alle previsioni (il consulente umano che serve per un confronto) ha effettuato una scelta simile, per quanto riguarda le estremità della giornata, ma ha scelto un diverso orario intermedio.

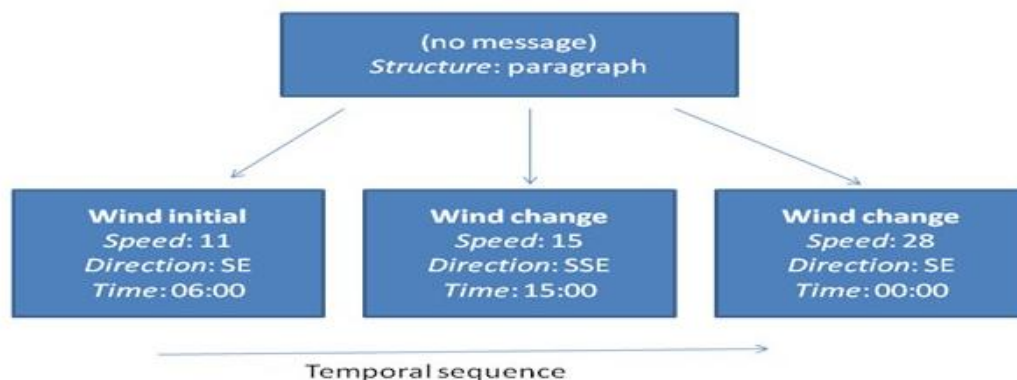


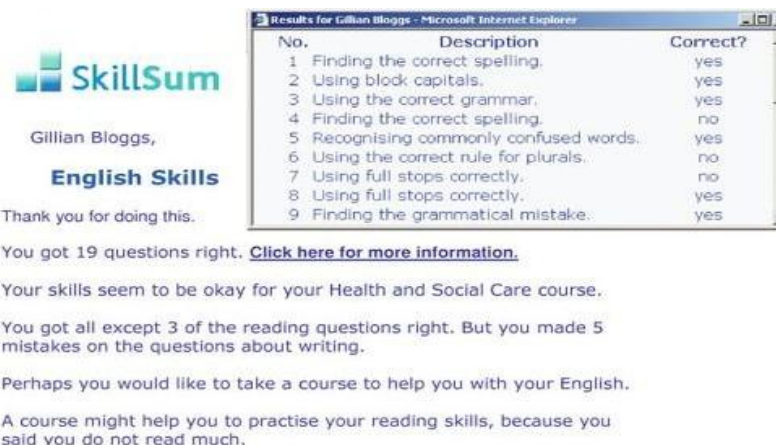
Fig. 26 - Esempio di Document Plan eseguito da *SumTime*

Altri tipi di scelte riguardano elementi strettamente linguistici: le parole, la sintassi delle varie frasi. Un esempio di scelta delle giuste parole viene evidenziato nell'esempio allorquando il sistema deve comunicare l'orario 00:00: mentre l'esperto umano ha utilizzato la frase “later”, SumTime ha preferito l'uso dell'espressione “by midnight”

Un'altra importante categoria di scelte riguarda l'ordine delle parole e la loro forma (*realization*). Un esempio di tale categoria è effettuato nel presentare i verbi con forme di participio presente (in ‘ing’); tale scelta in questo caso è basata sulle convenzioni particolari del tipo di linguaggio (quello “meteorologico”) che abbiamo preso in considerazione.

Il nostro secondo esempio, SkillSum, realizzato da Sandra Williams e Ehud Reiter nel 2008⁸² genera riassunti di prestazioni scolastiche riguardanti determinati studenti.

⁸² SkillSum fu realizzato da Williams e Reiter dal 2003 al 2005 presso l'Università di Aberdeen. Si consulti l'indirizzo <http://www.abdn.ac.uk/ncs/departments/computing-science/skillsum-314.php>



No.	Description	Correct?
1	Finding the correct spelling.	yes
2	Using block capitals.	yes
3	Using the correct grammar.	yes
4	Finding the correct spelling.	no
5	Recognising commonly confused words.	yes
6	Using the correct rule for plurals.	no
7	Using full stops correctly.	no
8	Using full stops correctly.	yes
9	Finding the grammatical mistake.	yes

Thank you for doing this.

You got 19 questions right. [Click here for more information.](#)

Your skills seem to be okay for your Health and Social Care course.

You got all except 3 of the reading questions right. But you made 5 mistakes on the questions about writing.

Perhaps you would like to take a course to help you with your English.

A course might help you to practise your reading skills, because you said you do not read much.

Fig. 27 - Esempio di output fornito da SkillSum

Gli utenti di tale sistema sono gli stessi studenti e altri addetti all'attività di apprendimento in una specifica materia per la quale sono stati predisposti dei test attitudinali. L'input fornito a SkillSum consiste in alcune informazioni riferite all'utente, e in un prospetto delle sue attitudini già stabilite: inoltre viene fornito un documento con una serie di risposte relative a un insieme di domande a scelta multipla come nel seguente esempio:

One of these sentence has a word which is wrong.
Click on it.

- 1) He was walking to the canteen when he slipped on a wet floor.
- 2) I was walking to the canteen when I slipped on a wet floor
- 3) They was walking to the canteen when they slipped on a wet floor.

L'output di SkillSum è un testo che riassume la prestazione dell'utente nell'ambito di uno dei test attitudinali:

Thank you for doing this.
You got 19 questions right. [Click here for more information.](#)
Your skills seem to be OK for your Health and Social care course.
You got all except 3 of the reading questions right. But you made 5 mistakes on the questions about writing.
Perhaps you would like to take a course to help you with your English.
A course might help you to practice your reading skills, because you said you do not read much.
[Click here for Key Skills at XXX College.](#)

Sebbene il contesto di utilizzo di SkillSum sia completamente diverso da quello di SumTime, i criteri di sviluppo di entrambi i sistemi sono analoghi, e analoghe sono le scelte che essi sono invitati a effettuare nell'ambito della generazione dei testi finali. Anche SkillSum ad esempio deve decidere; per esempio il *document planner*⁸³ deve scegliere quanto dettagliatamente esplicitare i vari errori dell'utente. Il cosiddetto *microplanner* del sistema (una sorta di programma intermedio) deve inoltre specificare come esprimere

⁸³ I vari componenti costitutivi di un sistema di generazione testuale (document planner, microplanner e realizer) saranno definiti in dettaglio nel prossimo paragrafo.

linguisticamente tale informazione; per esempio, deve scegliere se dire: “*You got 19 questions right*” oppure “*You got 19 questions correct*”.

Infine il *realizer* (il programma che determina la realizzazione “ultima” del testo) deve effettuare le proprie scelte di basso livello come ad esempio utilizzare nella frase finale la forma “questions” anziché “question” (nel senso che deve riconoscere l’entità “19” come plurale).

Esistono numerosi altri esempi di sistemi di generazione del linguaggio naturale,⁸⁴ come abbiamo detto, applicati ai contesti più differenti, ma in sostanza dobbiamo ammettere che nonostante tali favorevoli premesse, questi sistemi vengono utilizzati soltanto di rado nelle applicazioni reali, e la loro utilità può essere messa tuttora in discussione: lo stesso SumTime fu utilizzato soltanto per pochi anni, sebbene alcuni studi riscontrarono come gli utenti del sistema di previsione, nel valutare le due versioni, preferivano i resoconti effettuati da SumTime rispetto a quelli generati da esperti umani.

NLG Tasks

L’illustrazione 28 descrive a questo punto un semplice schema dell’architettura con la quale vengono generalmente costruiti i sistemi di generazione testuale; distinguiamo i due componenti essenziali: il *Discourse Planner*, il *Microplanner* e il *Surface Realizer*.

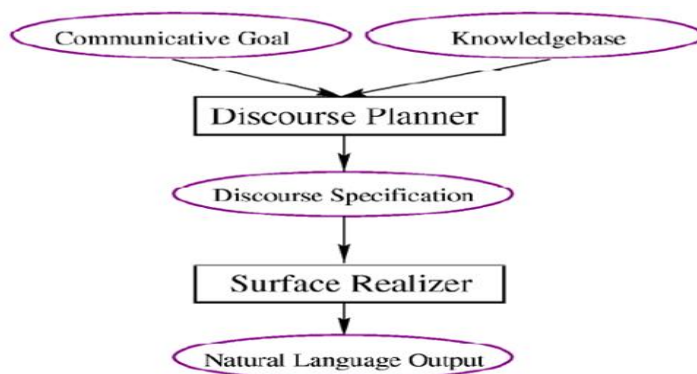


Fig. 28 - Architettura di un sistema NLG

Il Document Planner

Il Document Planner decide quali informazioni comunicare nel testo (*content determination*), e come queste informazioni dovranno essere organizzate (*document structuring*).

Questo componente si sviluppa in base a una ben definita meta comunicativa e, da una prospettiva del software, il suo input risulta l’input dell’intero sistema di generazione del testo; per esempio i dati numerici riferiti alle condizioni atmosferiche in SumTime, e le risposte alle domande dei test in SkillSum. L’output del document planner risulta tipicamente un insieme di messaggi strutturati. I messaggi sono raggruppamenti di informazioni (estratte dai dati d’ingresso), che possono essere espresse indistintamente in frasi che servono da prototipo.

⁸⁴ Esaustivo risulta anche l’elenco all’indirizzo: <http://www.fb10.uni-bremen.de/anglistik/langpro/NLG-table/nlg-table-system-sort.html>

In SumTime la struttura di un documento deriva dal modo di selezionare un ristretto numero di valori a partire dai dati in possesso: ricordiamo, una scelta delle condizioni iniziali e finali preludono a una interpolazione tra questi valori: è possibile effettuare una interpolazione tra tali valori minimizzando l'errore implicito che deriva dall'interpolazione, secondo criteri esclusivamente statistici.

La struttura di un documento richiesto in SumTime risulta molto semplice: i messaggi sono sempre collegati da una relazione che specifica una sequenza temporale.

Il sistema STOP,⁸⁵ che genera dei messaggi riguardanti le problematiche legate al fumo rivolti a dei fumatori che vogliono smettere, è basato invece sul materiale fornito dalla compilazione di determinati questionari: occorre una pianificazione di modelli psicologici e linguistici, per scegliere quale informazioni dovrebbero essere date ai fumatori, in base al loro desiderio di smettere di fumare e alle loro aspettative.

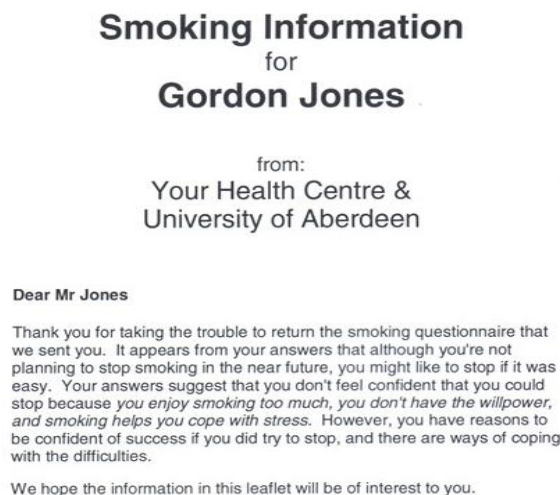


Fig. 29 - Pagina di una lettera generata dal sistema STOP.

Sfortunatamente, è difficile effettuare delle generalizzazioni riguardo l'attività di pianificazione iniziale: ogni sistema utilizza un proprio approccio e gli algoritmi sono a volte completamente differenti.

Un approccio alternativo è quello di cercare di imitare ciò che gli scrittori "umani" fanno, a volte producendo materiale senza esplicitamente modellare o ragionare ciò che infine risulterà utile per il fruitore finale, il comune lettore. In ogni caso il più grande problema riguardo il tipo di approccio da scegliere è la "sparsità" tra i dati e la loro incompletezza.

Tale tipo di analisi spesso richiede un corpus esaustivo di dati e allo stesso tempo testi annotati manualmente.

Microplanning

L'attività di microplanning decide come le informazioni sono espresse indistintamente nel testo generato. Questo processo richiede in genere le seguenti scelte:

⁸⁵ STOP: si può consultare il sito <http://inf.abdn.ac.uk/research/stop/>

- 1) scelte lessicali: si scelgono le singole parole da utilizzare per esprimere i vari concetti e descrivere i vari dati;
- 2) scelte dei riferimenti: si scelgono le espressioni più opportune per identificare le entità presenti nei vari domini e le loro relazioni;
- 3) scelte sintattiche: le strutture sintattiche con le quali generare le varie frasi;
- 4) scelte di aggregazione: si decide quanti messaggi verranno espressi in ciascuna frase.

Ad esempio il microplanner del sistema SkillSum nel caso debba riferire a un utente che ha fornito venti risposte corrette, indice di una ottima prestazione, potrebbe avere le seguenti possibilità di scelta:

- a) You scored 20, which is very good.
- b) You scored 20. This is very good.
- c) You got 20 answers right! Excellent!
- d) Excellent, you got 20 answers right!
- e) 20 questions were answered correctly: this is a very good score.

Ciascuna di queste scelte implica naturalmente una particolare impostazione del lessico, o dell'ordine delle parole, o dell'organizzazione sintattica.

In ogni caso, mentre l'input del microplanner è il documento creato durante la fase di pianificazione, l'output consiste in una serie di specifiche riguardanti il testo finale: tali specifiche possono essere definite in una sorta di struttura ad albero in cui i nodi interni specificano le strutture del documento (per esempio, i soliti paragrafi), e in cui i nodi marginali specificano le strutture sintattiche delle frasi.

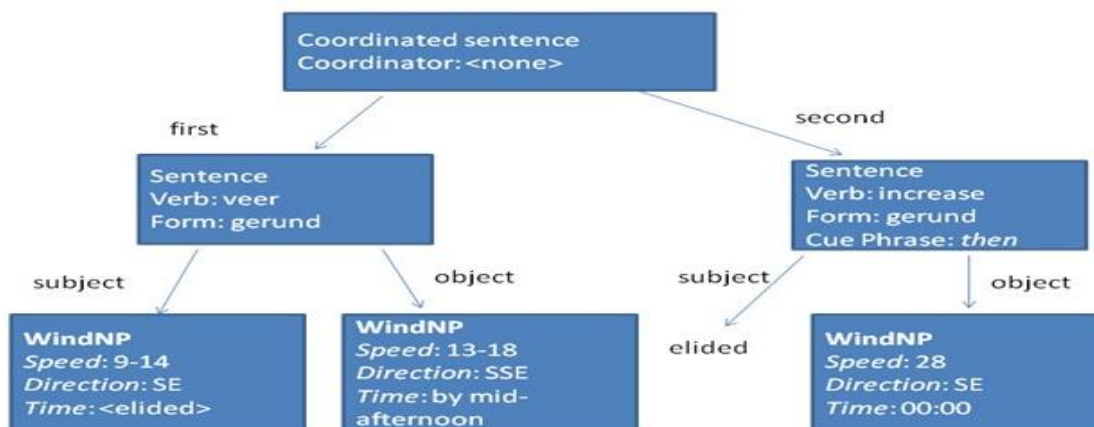


Fig. 30 - Esempio della struttura profonda di un testo prodotto da SumTime

Un esempio di struttura sintattica profonda è quello mostrato nella figura appena presentata: tale schema risulta difatti una successiva elaborazione dello schema di figura 26. Possiamo notare la struttura WindNP che comunica i valori dei vari parametri: velocità del vento, direzione, e orario.

Con tali strutture è piuttosto semplice realizzare dei messaggi costruiti con criteri simili ai cosiddetti canned texts.

Ad esempio riguardo la seconda struttura, una frase generabile risulterà:

By Mid-afternoon the wind direction will be *SSE* with a speed of *13,18* meters per second

A partire da tre file di testo contenenti le espressioni intermedie e le loro varianti è possibile costruire un algoritmo in grado di produrre una frase basata sulla struttura WindNP. Il seguente listato definisce un modulo per la produzione di testi secondo il modello di SumTime.

```
public ref class WindNP
{
public:
String ^ sDirection;
String^ sSpeed;
String^ sTime;

public WindNP(String^ s1,String^ s2,String^ s3)
{
sDirection=s1;
sSpeed=s2;
sTime=s3;
}

}

private: System::Void button3_Click(System::Object^ sender, System::EventArgs^ e) {
WindNP^ w1=gcnew WindNP("SSE","13-18","by mid-afternoon");

String^ s1, ^s2,^s3,^s4,^s5,^s6;

s1=w1->sTime;
s2=ASFiles::sentFile(L"texts\\conn1.txt");
s3=w1->sDirection;
s4=ASFiles::sentFile(L"texts\\conn2.txt");
s5=w1->sSpeed;
s4=ASFiles::sentFile(L"texts\\conclusionone.txt");

String^ sResult= s1+s2+s3+s4+s5+s6;
MessageBox::Show(sResult);

}
```

Riguarda un esempio di scelta lessicale la maniera di comunicare l'orario 15:00, o anche di comunicare la velocità con il valore 15, a partire dal riferimento 13-18

Un esempio invece di scelta preferenziale è mostrata da SkillSum per definire il nome del corso frequentato, o oggetto dell'attenzione: possiamo scegliere tra le varianti “*your Health*

and Social Care course.” o anche “your course”, oppure “*your Health and Social Care course at Aberdeen College*”;

Ovviamente nei due sistemi considerati le scelte effettuate dalla micro pianificazione raggiunge differenti livelli di sofisticatezza. Alcune scelte sono indotte dal sotto linguaggio utilizzato: per esempio SumTime utilizza sempre le forme del gerundio dei verbi per indicare le varie direzioni e velocità dei venti, poiché questa è la maniera con la quale convenzionalmente tali testi sono trattati. Alcune scelte richiedono un basso livello di elaborazione ma sono basate su sostanziali analisi linguistiche o psicolinguistiche dei dati.

Un’interessante osservazione nell’ambito di questi lavori riguarda il fatto che i testi prodotti da parlanti umani non necessariamente sono ideali per la comprensione degli ascoltatori o degli eventuali lettori. Difatti gli algoritmi generanti espressioni preferenziali di alta qualità non possono essere basati essenzialmente sull’analisi dei vari corpora a disposizione, o su altri studi di riproduzione del linguaggio umano: essi devono essere inseriti nell’ambito di teorizzazioni a un livello più elevato e come sempre più ampio di quello umanizzante con il quale, generalmente, consideriamo l’NLG.

Surface Realization

Il componente che realizza tale funzione produce sequenze di parole specifiche in base alle restrizioni e alle regole definite nel lessico e nella grammatica.

Tale componente trasforma il proprio input (frammenti di frasi che contengono le varie informazioni essenziali) e costruisce frasi grammaticalmente accettabili in base al lessico di riferimento.

Esistono due approcci essenziali per realizzare questo stadio: il primo basato sulla cosiddetta *Systemic Functional Grammar*, il secondo sulla *Functional Unification Grammar*.

Ad esempio si pensi di dover realizzare una frase molto comune per un utente che lavora al computer

The system will save the document

Questa frase potrà essere generata a partire da una semplice indicazione fornita dal *Text Planner* riguardante la necessità di salvare un documento

<action>save</action><document>title</document>

In altri casi potranno essere definite ulteriori specifiche riguardanti le forme grammaticali dell’espressione finale (ad esempio il tempo verbale) e anche particolari termini lessicali o altri aspetti sintattici come la forma passiva o attiva della frase.

<action>save</action><document>title</document><tense>present</tense>

Esistono due possibili approcci alla realizzazione del Realizer: la *Systemic Functional Grammar* e la *Functional Unification Grammar*.

Per *Systemic Functional Grammar* intendiamo un branca della linguistica, ideata da Michael Halliday, nella quale il linguaggio viene visto come una architettura multifunzionale che riflette la natura multidimensionale dell'esperienza e delle relazioni umane.⁸⁶

Tale approccio risulta piuttosto adatto alla generazione di testi ed è stato ampiamente utilizzato nell'ambito dell'NLG.

In tale Grammatica le frasi sono rappresentate come una collezione di funzioni definite in vere e proprie mappe.

	<i>The system</i>	<i>will</i>	<i>save</i>	<i>the document</i>
Mood	subject	finite	predicator	object
Transitivity	actor	process		goal
Theme	theme		rheme	

Fig. 31 - Gerarchia delle funzioni rappresentative.

Osserviamo la presenza di tre strati ciascuno con delle funzioni rappresentative specifiche in grado di definire lo svolgersi dell'azione di partenza nei suoi contenuti e nelle forme.

Nell'approccio della *Functional Unification Grammar* si utilizzano regole di unificazione per manipolare e sviluppare le varie strutture di caratteristiche

In questo sistema gli oggetti sono rappresentati da descrizioni funzionali (*functional description* o FD).

Una FD è una espressione booleiana riguardante un insieme di caratteristiche; la seguente FD descrive tutte le frasi il cui soggetto è una frase nominale, singolare al caso nominativo o accusativo:

$$\left[\begin{array}{l} \text{Cat} = \text{S} \\ \text{Subj} = \left[\begin{array}{l} \text{Cat} = \text{NP} \\ \text{Num} = \text{Sing} \\ \left\{ \begin{array}{l} \text{Case} = \text{Nom} \\ \text{Case} = \text{Acc} \end{array} \right\} \end{array} \right] \end{array} \right]$$

Fig. 32 - Esempio di una semplice *Descrizione funzionale*.

Come è ovvio, strutture più complesse possono essere formate aggregando strutture più semplici:

⁸⁶ Halliday Michael, *Spoken and Written Language*, Deakin University Press, Victoria (1985).

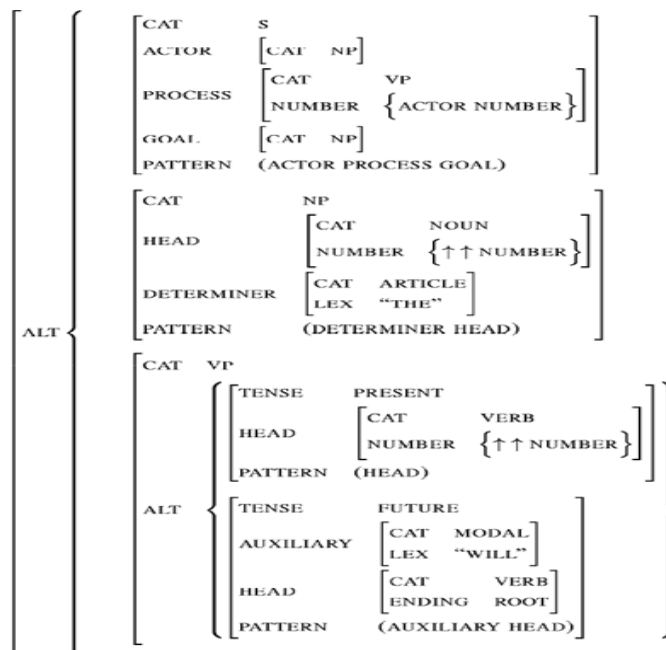


Fig. 33 - Un modello di grammatica FUF

Si suppone difatti l'esistenza di livelli multipli di strutture che vengono utilizzate in modo ricorsivo. Secondo tale sistema la frase di partenza potrà essere rappresentata in una struttura del tipo

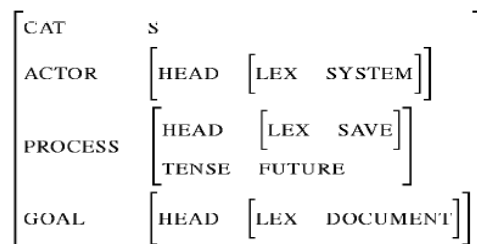


Fig. 34 - Rappresentazione della struttura descritta anche in ambito della *Systemic Functional Grammar*

Nella descrizione presentata ritroviamo l'esistenza di un attore, cioè il sistema, di una meta, il documento, e di un processo che definisce ciò che deve svolgere l'attore per ottenere la meta. La struttura presentata può essere ampliata da ulteriori informazioni come nella seguente nuova struttura:

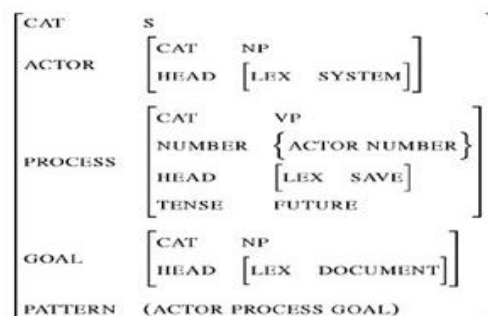


Fig. 35 - Descrizione funzionale: riguardo la caratteristica PROCESS viene aggiunta la specificazione del numero degli attori NUMBER. Viene aggiunta inoltre anche la caratteristica PATTERN

Strutture analoghe a quelle finora indicate, sono presenti in *simplenlg* una libreria JAVA, progettata per facilitare la generazione testuale nella fase di *realisation*. Sviluppata originariamente all'università di Aberdeen è costituita da un sistema lessicale e morfologico in grado di calcolare le varie realizzazioni morfologiche di una frase in base a determinati parametri.

Si consideri il codice seguente:

```
import simplenlg.framework.*;
import simplenlg.lexicon.*;
import simplenlg.realiser.english.*;
import simplenlg.phrasespec.*;
import simplenlg.features.*;

public class SampleText {

public static void main(String[] args) {
    // TODO Auto-generated method stub
    //System.out.println("hello there");
    Lexicon lexicon = new XMLLexicon();//XMLLexicon("my-lexicon.xml");
    NLGFactory nlgFactory = new NLGFactory(lexicon);

    Realiser realiser = new Realiser(lexicon);

    SPhraseSpec p1 = nlgFactory.createClause();
    p1.setSubject("Harry");
    p1.setVerb("chase");
    p1.setObject("the monkey");
    String output1 = realiser.realiseSentence(p1);
    System.out.println(output1);
    p1.setFeature(Feature.TENSE, Tense.PAST);
    String output2 = realiser.realiseSentence(p1);
    System.out.println(output2);

    p1.setFeature(Feature.TENSE, Tense.FUTURE);
    String output3 = realiser.realiseSentence(p1);
    System.out.println(output3);
}
}
```

A partire da una frase definita soltanto attraverso i suoi elementi principali: un soggetto, un verbo e un oggetto è possibile ricostruire (uno dei tanti possibili esempi) la sua versione al passato e al futuro, attraverso le istruzioni evidenziate. L'output ottenuto è il seguente:

```
Tasks Console
<terminated> SampleText [Java Application] C:\Programmi\Java\jre6\bin\javaw.exe (29/nov/2013 14:16:54)
Harry chases the monkey.
Harry chased the monkey.
Harry will chase the monkey.
```

Fig. 36 - Schermata dell'output prodotto dal breve programma *SampleText*

Simplenlg è attualmente disponibile per l'inglese e il francese, mentre una versione per il tedesco è in fase di sviluppo.

La generazione testuale di riassunti

Come sappiamo, riassumere un testo o qualsiasi altra forma di rappresentazione multimediale (un film, un romanzo, il resoconto di una assemblea, gli atti di un convegno, o anche il contenuto di pagine internet) significa individuare i temi principali dell'oggetto da riassumere ed esprimerli in una nuova, abbreviata, rappresentazione. Nel nostro discorso ci concentreremo soprattutto nel trattare il riassunto di testi scritti, una disciplina, fondamentale nella realtà contemporanea, caratterizzata come abbiamo più volte osservato in una quasi iperbolica sovrabbondanza delle informazioni. Difatti in qualsiasi attività professionale siamo costretti a ricercare le informazioni che al momento più interessano, e nella quantità che riusciamo generalmente a gestire, spesse volte soltanto a livello di memoria a "breve" termine.

I problemi specifici riguardanti la generazione di riassunti risultano, per altro, distinti da quelli della generazione, così come l'abbiamo finora intesa. I riassunti ideali hanno difatti un obiettivo concreto: fornire il maggior numero di informazioni nel minor spazio possibile. Ciò significa, innanzi tutto, scegliere quelle strutture sintattiche che possono riprodurre l'informazione in modo conciso: ad esempio con il frequente ricorso a modificatori dei nomi e dei verbi, alle congiunzioni, alle ellissi; in genere con una composizione di frasi avente un minore numero di parole rispetto alle frasi normali.

I primi sistemi per il riassunto di testi si basarono esclusivamente su criteri statistici mentre solo in un secondo tempo vennero sviluppate delle applicazioni che prendevano in considerazione l'attività riassuntiva dal punto di vista cognitivo, così come gli esseri umani la concepiscono.

Il procedimento riassuntivo si svolge in sostanza secondo le seguenti fasi: nella prima il contenuto del testo originale viene classificato secondo una rappresentazione che ne identifichi l'importanza delle varie parti secondo determinati criteri circostanziali. La rappresentazione delle informazioni viene poi, di norma, accorciata, sottraendo le parti che sono meno frequenti o che hanno una presenza minore nel documento. La formulazione delle nuove frasi viene poi realizzata in modo da far confluire più informazioni possibili con il minor numero di parole. Successivamente si produce una ulteriore riduzione che fornisce la rappresentazione esterna finale del testo considerato

Modello di Sistema riassuntivo

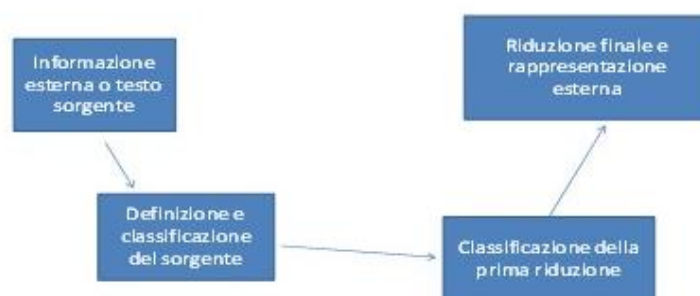


Fig. 37 - Modello generale di sistema riassuntivo.

Indicazioni pratiche e criteri di procedimento.

Un sistema per la realizzazione può essere costruito su semplici regole di facile attuazione.

1) Si possono considerare le varie lunghezze delle frasi: a frasi che hanno una lunghezza superiore, ad esempio, alle 7 parole, viene attribuito un valore negativo poiché difficilmente inseribili in un riassunto.

2) Si cerca di individuare degli speciali indicatori stabiliti inizialmente: frasi che contengono tali indicatori vengono valutate con parametri positivi. Tali indicatori speciali possono essere espressioni come “In conclusione...”, o “Abbiamo scoperto che...”.

3) Si analizza poi la struttura e la composizione delle frasi: le prime dieci frasi e le ultime dieci vengono definite con valori positivi. Ovviamente si presuppone che all’inizio e alla fine di ogni testo si premettano e si riassumano già di per sé, di solito, i temi principali trattati nel testo.

4) Si individuano determinate parole chiave definibili in base alla frequenza con cui le parole si trovano nel testo (escluse ovviamente le parole presenti nell’elenco delle stopword) oppure in base a un criterio stabilito di volta in volta esaminando il contesto in cui si è realizzato lo scritto da riassumere.

5) Acronimi. Le frasi contenenti degli acronimi (ad esempio IBM) vengono definite con un valore positivo.

Attraverso l’applicazione di tali criteri, il valore di ogni frase è la media pesata dei singoli valori ottenuti nella prima fase e successivamente scelte come input per una successiva elaborazione. I criteri di scelta del numero di frasi totali sarà ovviamente correlato a dei parametri scelti inizialmente o legati alla dimensione del testo di partenza.⁸⁷

⁸⁷ Online, tra i tanti è disponibile un text Summarizer all’indirizzo <http://libots.sourceforge.net/>

7. La conoscenza comune

Nell'ambito della costruzione di una interfaccia dialogante è essenziale definire una base di conoscenza del mondo di riferimento su cui è costruita l'interfaccia stessa. Il sistema dovrà essere in grado di conoscere una gamma di argomentazioni difficili da concettualizzare formalmente. È forse paradossale, ma per gli scienziati operanti nel campo dell'IA molti argomenti facili da comprendere per esseri umani di appena dieci anni risultano più refrattari ai tradizionali metodi di comprensione dell'AI stessa, di quanto non lo siano gli argomenti oltremodo più complessi.

I fisici sono capaci di descrivere fenomeni naturali in modo esatto e dettagliato con le equazioni d'onda, la teoria della relatività e altri costrutti matematici, mentre i ricercatori di IA discutono ancora sul modo migliore di rappresentare, ad esempio, il semplice evento di un liquido che riempie una tazza e cade se la tazza viene capovolta. Le caratterizzazioni altamente teoriche inventate dai fisici e dai matematici sembrano più facili da formalizzare dei concetti più semplici.

Consideriamo solo alcune delle cose che un bambino di dieci anni conosce:

Se si lascia andare un oggetto questo cadrà.
Le persone non esistono prima di essere nate.
I pesci vivono in acqua e muoiono se tolti da essa.
Il pane e il latte si comprano in negozi di generi alimentari.
Le persone dormono nei letti.

Una conoscenza di questo tipo è solitamente chiamata *conoscenza comune*.

In genere, la conoscenza su un argomento è diffusa a vari livelli, da quello tipico della persona della strada, a quello dello specialista.

Le teorie scientifiche intorno al 500 a. c., erano poco più che attente formulazioni verbali di osservazioni quotidiane effettuate dalle persone comuni: per alcuni la terra era piatta, gli oggetti cadevano a terra perchè quello era il "loro luogo naturale" e le malattie degli uomini erano causate da una varietà di pittoresche "influenze".

Quando la conoscenza era scarsa, i gradi di separazione tra la conoscenza comune e la conoscenza scientifica avanzata erano relativamente ridotti. La conoscenza comune era (ed è ancora) adeguata per molte delle cose che gli uomini vogliono fare; difatti la conoscenza scientifica si separò gradualmente dalla conoscenza comune quando gli uomini cercarono descrizioni più precise del mondo.

Perfino oggi, molte delle attività che vorremmo fossero svolte da un robot potrebbero basarsi sulla scienza medievale, se non preistorica. Comuni reazioni e deduzioni logiche che spesso l'uomo attua sin dagli albori, non richiedono una conoscenza avanzata: per esempio mettersi al riparo in caso di pioggia (senza la necessità di conoscere le leggi che regolano i fenomeni atmosferici, i sistemi di bassa e alta pressione ecc.), fare attenzione a non versare il caffè dalla tazza portandolo alla bocca (non è necessario conoscere l'idrodinamica avanzata), inserire la spina di un caricatore quando la batteria è appunto scarica (senza conoscere le teorie elettriche o elettrochimiche).

Per certi compiti è indubbiamente richiesta la semplice conoscenza in possesso di un bambino, eppure tale conoscenza risulta tuttora di difficile rappresentazione e gestione.

I ricercatori di IA hanno chiamato tale conoscenza del mondo, conoscenza *naive*, anche se contrapponendo la conoscenza *naive* alla conoscenza esperta si dovrebbe precisare che questi termini non sono deterministicamente definiti, ma costituiscono due regioni contigue che spesse volte si compenetrano l'un l'altra.

Difatti, attività differenti richiedono livelli differenti di conoscenza lungo questo continuum: prendendo ad esempio la fisica, un robot avrebbe bisogno di conoscere soltanto le proprietà semplici degli oggetti al fine di interagire in una realtà specifica: si pensi all'esperimento del chatbot SHRDLU e al movimento dei blocchi o delle varie forme analoghe all'interno di una stanza.

La fisica di senso comune per un tale robot tratterebbe fatti specifici: due blocchi non possono trovarsi nello stesso luogo, un blocco deve essere sorretto dal tavolo o da un altro blocco e così via.

Necessità più complesse implicherebbero rappresentazioni più articolate e leggi fisiche più analitiche, e quindi la nozione stessa di conoscenza comune deve essere adattabile alle esigenze più svariate.

Uno dei tentativi più interessanti che cerca di perseguire questi scopi è quello del sistema Cyc⁸⁸ che sfrutta la costruzione di una base di conoscenza comune attraverso annotazioni manuali. Cyc, realizzato attraverso il contributo di Doug Lenat e dei suoi collaboratori si sviluppa a partire da un approccio per certi versi di *forza bruta*⁸⁹ con la conoscenza da rappresentare. Nel 1990, gli autori di CYC avevano asserito:

Forse la verità più dura da affrontare, quella a cui l'IA sta cercando di sfuggire da 34 anni, è che non vi è probabilmente un modo elegante e facile per ottenere questa immensa base di conoscenza. Piuttosto il grosso dell'impresa dovrà (almeno inizialmente) essere un'introduzione manuale di asserzioni, una dopo l'altra.⁹⁰

In definitiva, la conoscenza comune sembra non avere frontiere ben definite che ci consentano di catturare parti di essa indipendentemente da altre. Inoltre tale conoscenza non può essere sempre ed esclusivamente catturata da frasi dichiarative. Descrivere le forme e altri oggetti fisici complessi per mezzo di frasi è difficile: ad esempio, può un volto umano essere descritto a parole in modo che possa essere riconosciuto da un'altra persona che non lo ha mai visto prima? Che parole usiamo per catturare la definizione di un'auto, di un treno, di un mezzo di trasporto su ruote? Sono soddisfacenti le informazioni strutturate tratte da una ontologia costruita con questi criteri?

⁸⁸ Il sito del sistema Cyc è all'indirizzo <http://www.cyc.com/>

⁸⁹ Il termine, utilizzato in particolare dagli hacker, indica un approccio che valuta tutte le varie possibilità di una scelta senza particolari criteri logici o deduttivi: ad esempio per individuare una password vengono provate in sequenza tutte le combinazioni di caratteri valide per quella password.

⁹⁰ Guha R. V., Lenat D., "Cyc: A Midterm Report", *AI Magazine*, pp. 33-59 (1990).

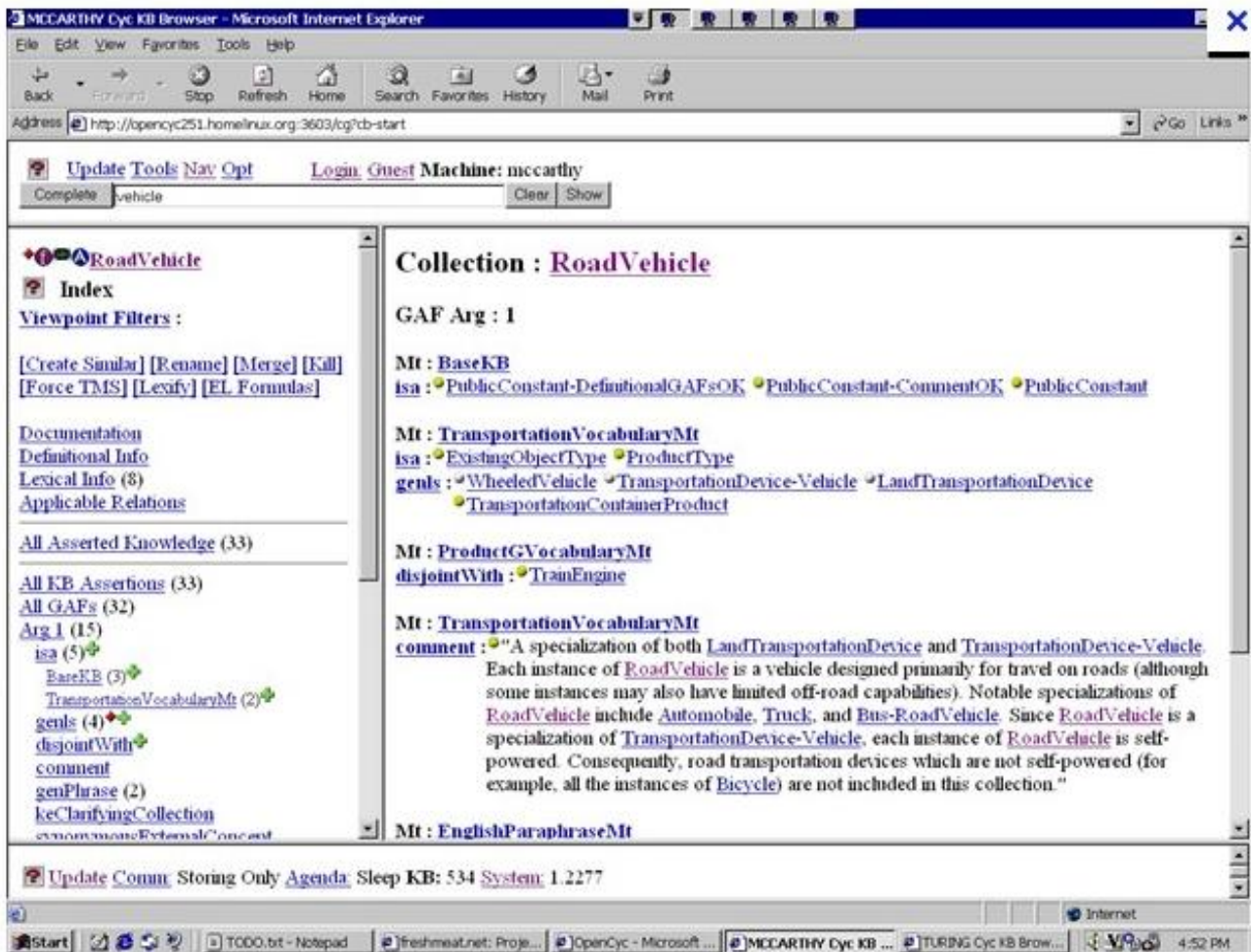


Fig. 38 - Esempio di concettualizzazione del termine “Road Vehicle” attraverso il sistema OpenCyc.

Connessa alla difficoltà di catturare la conoscenza in una frase dichiarativa è la difficoltà legata alla approssimazione con cui molte frasi utilizzate per descrivere il mondo operano effettivamente. In particolare, le frasi universali (ossia quelle della forma “Tutte le x sono y ”) sono raramente valide, a meno che non risultino semplici definizioni in contesti specifici come ad esempio quello della matematica: molta conoscenza è difatti approssimativa e le valutazioni che se ne possono fare hanno *sempre* un carattere probabilistico. Anche la semplice asserzione “il sole sorgerà domani”, a rigor di logica ha il suo valore di probabilità.

Inoltre, se le leggi della statistica potrebbero venirci in soccorso per gestire la “non linearità” del mondo che ci circonda, e “trasformarlo” paradossalmente in un computo di valori numerici, non sempre risulta ben chiaro come concettualizzare e organizzare alcuni argomenti.

Dovremo pensare al tempo cronologico, ad esempio, come rappresentato dall’insieme dei numeri reali (cioè come un continuum di *istanti*), o immaginare il *passato* come una singola *linea nel tempo*, così come immagineremmo il futuro? Dal momento che il futuro non è definito quale soluzione sarebbe possibile, per formulare un modello, che includa tutte le eventualità di una ipotesi.

È necessario quindi riconoscere che concettualizzare gli aspetti fondamentali del mondo potrebbe risultare un buon punto di partenza per qualsiasi sviluppo nell'ambito dell'Intelligenza Artificiale e della comprensione del linguaggio naturale.

Il sistema Cyc rappresenta tale conoscenza secondo una progressione gerarchica che tiene presente le diverse caratteristiche del mondo che ci circonda. Innanzi tutto ne distingue gli *oggetti e i materiali*, quindi la loro collocazione nello *spazio*, e le loro proprietà fisiche. Individua i processi e gli eventi fisici e il trascorrere del tempo. La fig. 39 riassume in modo grafico una simile rappresentazione e le sue caratteristiche.

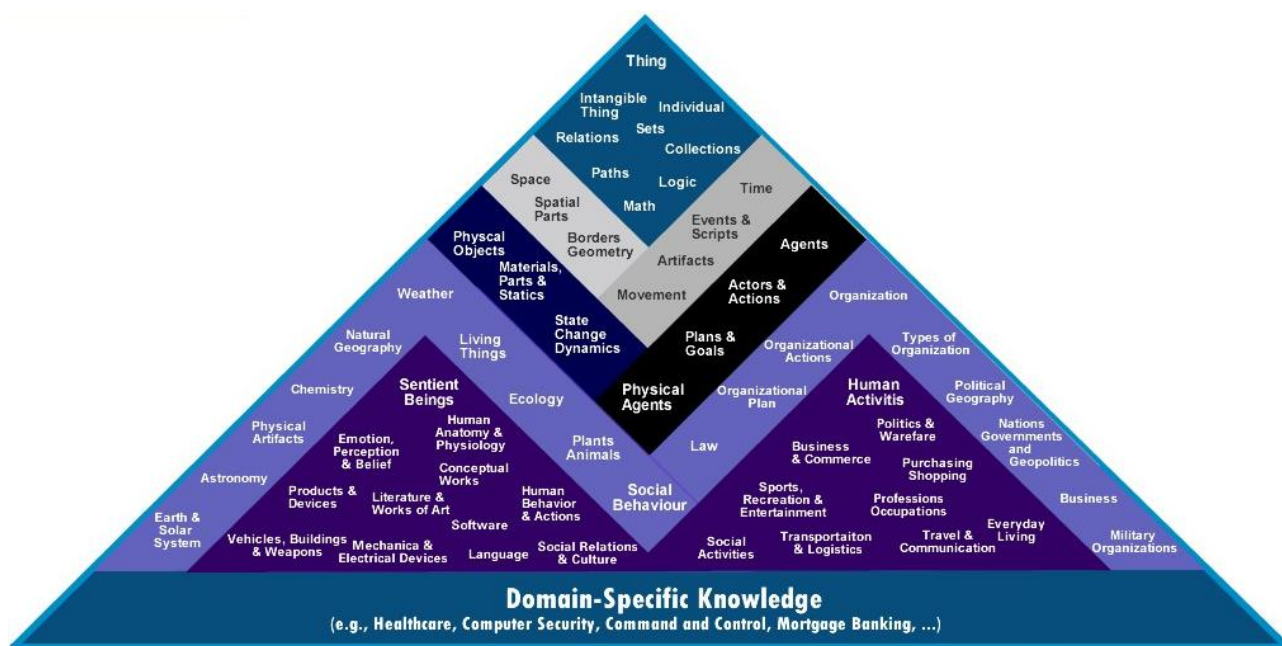


Fig. 39 - Una rappresentazione grafica della conoscenza strutturata nel sistema Cyc.

La struttura di questa base di conoscenza è tale che tutti i fatti presentati al suo interno sono collegati ad altri concetti. Per esempio l'ontologia definita dal sistema Cyc conosce all'interno del settore "musica" l'esistenza di centinaia di musicisti che non vengono considerati alla stregua di una semplice lista ma ciascuno di essi è associabile a un insieme di più di 80 categorie di riferimento. Per esempio l'elemento "Bruce Springsteen" non è una semplice stringa in una lista, ma possiamo sapere dai riferimenti ad esso collegati che è una persona, un professionista, un attore, e molte altre informazioni a loro volta connesse con ulteriori informazioni.

Nella rappresentazione di figura 39 possiamo individuare come l'entità fondamentale è denominata con la costante individuale *Cosa* (thing).

In Cyc vi sono diversi generi di *cose*: oggetti nel mondo, oggetti matematici, eventi e processi, e così via. Tali entità sono organizzate in una tassonomia che codifica implicitamente fatti della forma "X è un P, tutte le P sono Q, tutte le Q sono R", e così via

Le gerarchie tassonomiche possono essere codificate in vari tipi di strutture a carattere reticolare (reti semantiche) o basate su *frame*.

Un esempio di rete semantica può essere il seguente:

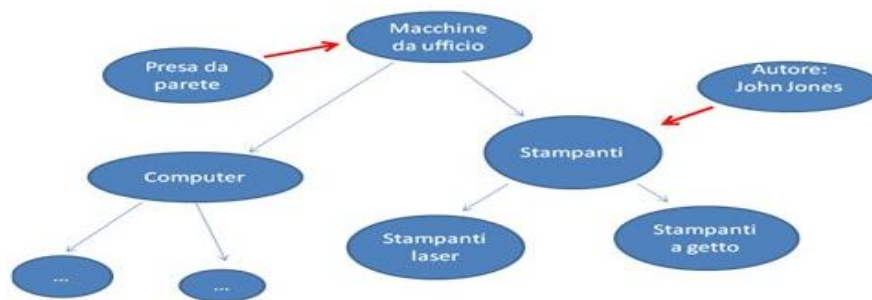


Fig. 40 - Esempio di rete semantica: i vari archi possono descrivere dei sottoinsiemi, delle funzioni, e così via...

La stessa rete semantica, sotto forma di frame, (ricordiamo che un frame ha un nome che lo definisce e che individua una lista di coppie attributo valore) è riprodotta nel seguente schema:

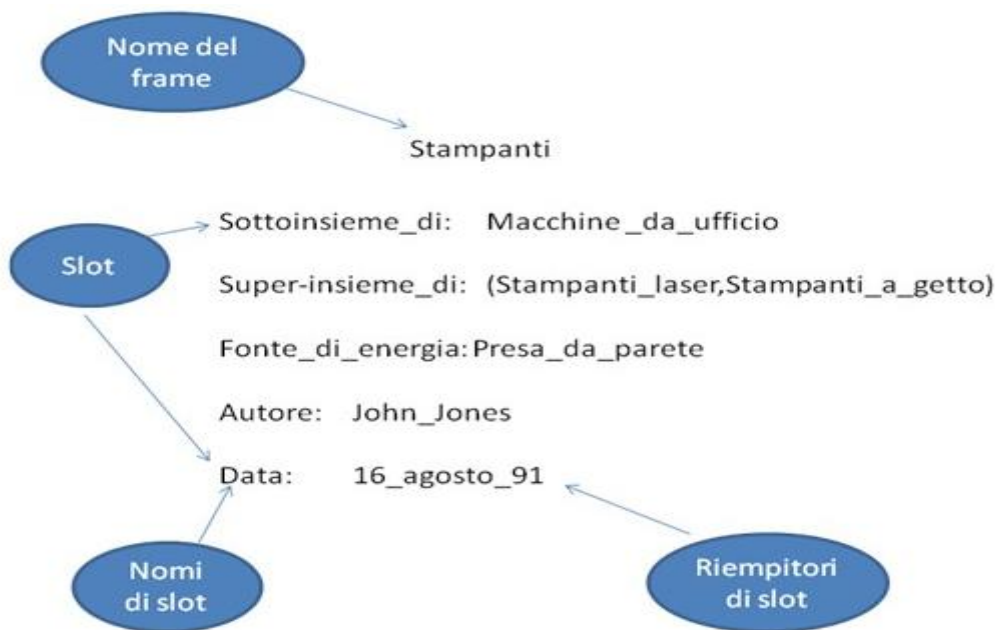


Fig. 41 - Frame del concetto *Stampanti*

Un altro esempio di sistema in grado di concettualizzare la conoscenza comune è offerto dal progetto *Open Mind Common Sense*.⁹¹

Tale progetto si propone di restringere il più possibile il ricorso ad annotazioni manuali della base di conoscenza ma sfruttando le risorse del web, collezionando (ovviamente sempre grazie all'ausilio di utenti umani) annotazioni di decine di migliaia di generici fatti.

⁹¹ Il progetto *Open Mind Common Sense* è presente in rete all'indirizzo <http://commons.media.mit.edu/en/>

Lo stratagemma utilizzato è quello di indurre i vari utenti e navigatori del Web a produrre le loro conoscenze partecipando a una serie di giochi disponibili online. Lo scopo annunciato dai creatori di Open Mind è quello di sviluppare un database di informazioni riguardanti la conoscenza comune del mondo, attraverso semplici frasi in inglese. Le frasi verranno elaborate dai server del sito messo a disposizione, analizzate e connesse con altri concetti individuando nuove informazioni.

Un esempio di parte dei risultati ottenuti con una ricerca sul termine “dog” è il seguente:

The screenshot shows the Open Mind Common Sense interface. At the top, it says "Open Mind Common Sense Explain your world." Below this is a navigation bar with "Home", "Add new knowledge", "Highest rated", and "Ad-hoc categories". A section titled "Similar concepts" lists "a person, a dog, a cat, something, a human, animals, children, a horse, a man, a computer". The main section is "Current knowledge", which displays a list of assertions with their authors and scores:

Assertion	Author	Score
→ An activity a dog can do is bark	by azidek	Score: 104
→ Dogs are mammals	by WWaotrlsdon	Score: 58
→ You are likely to find a dog in kennel	by Laserjoy	Score: 27
→ A golden retriever is a sort of dog	by breitmaulfrosch	Score: 21
→ An activity a dog can do is run	by nmeryash	Score: 21
→ A dog can guard a house	by Leika	Score: 21
→ a dog can be a pet	by jokev	Score: 19
→ A dog has four legs	by gary47	Score: 18

At the bottom, it says "Page 1 of 147 | Next | Last (1,176 total)".

Fig. 42 - Schermata tratta dal sito <http://commons.media.mit.edu/en/concept/dog/>

Nella fattispecie, un modulo della mia interfaccia dialogante nella versione inglese, presuppone la possibilità di ricercare tramite il sito eventuali informazioni. Il markup `action="commonsense"` gestisce tale eventualità:

```
<category>
<pattern><c>can I have a web search for*</c></pattern>
<template><c>only a minute, please</c><action>commonsense</action></template>
</category>
```

Il sistema individua nella pagina che presenta l’esito per la ricerca sul web (nel nostro esempio “dog”, ossia il valore attribuito al parametro “*”), ad esempio il seguente codice:

```
<td class="assertion_text">
  <a href="/en/assertion/96338/" class="predlink">&rarr;</a>
  An activity <a href="/en/concept/a%20dog/">a dog</a> can do is <a href="/en/concept/bark/">bark</a>
</td>
```

E ricava il contenuto interno del markup html: “An activity a dog can do is bark”

Ovviamente tra le tante proposte la scelta viene effettuata in base a criteri specifici dell'algoritmo da me realizzato.

Il ricorso alle basi di conoscenza presenti in rete caratterizzerà e per certi versi caratterizza lo sviluppo delle moderne interfacce dialoganti. La ricerca di una risposta a una propria richiesta verrà effettuata con modalità simili a quella con cui si utilizzano i vari motori di ricerca, ma come ampiamente detto, l'approccio sarà più umanizzato, proprio attraverso l'utilizzo del linguaggio naturale nelle sue forme più complete (non esclusivamente attraverso semplici parole chiave o una concatenazione di esse).

8. Interfacce dialoganti

La tradizione letteraria abbonda di episodi in cui figure inanimate acquistano il dono della parola e la facoltà di poter conversare attraverso il linguaggio naturale dell'uomo. Possiamo ancora ricordare la statua di Pigmalione nelle *Metamorfosi*⁹² di Ovidio, il burattino Pinocchio di Carlo Collodi⁹³, o anche nell'800 il già menzionato mostro creato dal dottor Frankenstein nell'omonimo romanzo di Mary Shelley.

In questo caso proprio la prerogativa del linguaggio risultava l'elemento fondamentale che conferiva umanità al *mostro*.

Un altro esempio rilevante è nell'odierno immaginario collettivo e popolare: lo specchio parlante che risponde alle domande della regina cattiva nella favola di Biancaneve ha tutte le caratteristiche di una sorta di interfaccia dialogante, qualcosa di inanimato con cui svolgere un dialogo o una conversazione.⁹⁴



Fig. 43 - La regina della celebre favola alle prese con le conversazioni allo specchio

REGINA

Mago dello specchio magico, sorgi dallo spazio
profondo, tra vento e oscurità io ti chiamo.
Parla! Mostrami il tuo volto!

SPECCHIO

Che vuoi conoscere mia Regina?

⁹² Ovidio, *Le metamorfosi*, X, 243-297.

⁹³ Collodi Carlo, *Le avventure di Pinocchio. Storia di un burattino* (1883). All'inizio della favola è addirittura il semplice pezzo di legno in possesso di Mastro Ciliegia a parlare e provocare un litigio tra Mastro Ciliegia e Geppetto.

⁹⁴ Ci sono diverse ipotesi sulle origini della storia di Biancaneve, che potrebbe essere ispirata a fatti realmente accaduti. Nel 1986 il ricercatore Karl-Heinz Barthels ipotizzò che Biancaneve sarebbe stata in realtà Maria Sophia Margaretha Catherina von Erthal, nata a Lohr nel 1725 e figlia di un importante magistrato e rappresentante del Principe Elettore tedesco. La nobile aveva perso la madre in età giovanile e suo padre si era risposato con Claudia Elisabeth von Reichenstein. Il castello dei von Erthal è tuttora un'attrazione turistica, e ai visitatori viene mostrato il cosiddetto "specchio parlante", che il padre di Maria Sophia avrebbe regalato alla matrigna: si tratta di un giocattolo acustico in voga nel '700, in grado di registrare e riprodurre le frasi pronunciate da chi si specchiava. Sarebbe alla base dello Specchio Magico della matrigna.

REGINA

Specchio, servo delle mie brame, chi è la più bella del reame?

SPECCHIO

Bella, tu sei bella oh mia Regina, ma attenta: al mondo una fanciulla c'è, vestita sol di stracci, poverina, ma ahimè, assai più bella è di te!

REGINA

Guai a lei! Dimmi il suo nome!

SPECCHIO

Ha la bocca di rose, e ha d'ebano i capelli, come neve è bianca.

REGINA

Biancaneve.

Oltre ai precedenti esempi, possiamo ricordare la leggenda secondo cui Michelangelo al termine della scultura del Mosè, si rivolse a quest'ultima, nella consapevolezza della sua perfezione, con la celebre esclamazione "Perché non parli?"

Sempre in Mary Shelley, a mio avviso proprio uno degli esempi più pregnanti, il mostro creato da Frankenstein esprime chiaramente la necessità di acquisire la competenza linguistica per conferire definitivamente umanità al proprio essere:

These thoughts exhilarated me, and led me to apply with fresh ardour to the acquiring the art of language. My organs were indeed harsh, but supple; and although my voice was very unlike the soft music of their tones, yet I pronounced such words as I understood with tolerable ease. It was as the ass and the lap-dog; yet surely the gentle ass, whose intentions were affectionate, although his manners were rude, deserved better treatment than blows and execration⁹⁵

Non dovremmo sorprenderci: il linguaggio è sempre stato l'elemento discriminante in grado di conferire la prerogativa dell'*umanità* a chi ne possedesse il dono: la *conversazione*, il *dialogo* costituiscono quindi, in ultima analisi la manifestazione più evidente e privilegiata dagli esseri umani per manifestare questa sorta di facoltà.

Di certo il primo approccio che noi tutti abbiamo con il linguaggio avviene proprio tramite una serie più o meno articolata ed evoluta di conversazioni, e la maggior parte degli esseri umani, in molteplici attività della nostra esistenza, dall'aprire un conto corrente, all'ordinare un pasto caldo al ristorante, proprio attraverso il dialogo e la conversazione, interagiscono col mondo circostante.

Ogni giorno, del resto, poniamo domande e otteniamo risposte a queste ultime. L'avvento di internet e la potenzialità delle risorse presenti in rete hanno conferito inoltre una ulteriore

⁹⁵ Shelley M., *Frankenstein*, p. 169

dimensione all'attività di dialogare, e la possibilità di dialogare con un computer è oggi giorno alla portata di sempre un maggior numero di persone.

L'avvento di internet è riuscito a cambiare la prospettiva di utilizzo dei sistemi di dialogo Uomo-Computer, in nuove ottiche di interazione, applicate su scala mondiale e in termini di tempo pressoché istantanei.

Tuttavia, anche se siamo abituati alla tradizionale ricerca basata su parole chiave (come in Google, Yahoo, e in altri motori di ricerca), evolvere tale approccio e gestire un vero e proprio dialogo alla scoperta di una informazione, potrebbe semplificare agli utenti la gestione delle informazioni online, in molteplici maniere.

Difatti, determinate domande non possono essere tradotte in una semplice lista di parole chiave. La domanda:

Quali nazioni visitò il Papa nel 1960?

non può essere risposta in riferimento alle parole chiave “nazioni”, “papa”, “visitò”, “1960”, poiché una ricerca basata esclusivamente su questi criteri troverà soltanto documenti (nel nostro caso pagine web) che contengono le parole suddette prese a volte singolarmente: potrà produrre cioè anche una serie di documenti incentrati sul token “1960” senza prendere in considerazione le varie intersezioni richieste con le altre parole. Sarebbe quindi necessario produrre un più complesso insieme di *parole chiave* allo scopo di raggiungere i nostri risultati, ma molto spesso gli utenti di Internet non sono in grado di formulare tali parole chiave, secondo una strategia davvero utile alle ricerche.

L'architettura di un moderno sistema di QA segue invece immancabilmente un processo di elaborazione che nell'utilizzo di documenti potenzialmente rilevanti, estrae le risposte più opportune (le risposte candidate) e tra queste ultime selezionerà la risposta principale e le eventuali alternative.

L'approccio iniziale si propone sostanzialmente l'elaborazione della domanda e si può scomporre in due fasi: la prima fase in grado di rilevare semplicemente la domanda stessa, la sua forma fisica; la seconda nel definire quali parti di essa sono funzionali per una ricerca e nel classificarle: *question typing*.

Un possibile schema di un QA System è il seguente:

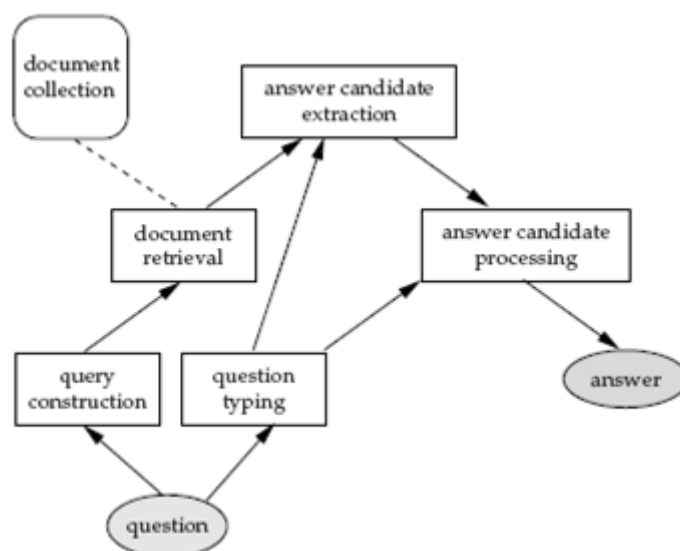


Fig. 44 - Basilare architettura di un Sistema QA.

Ad esempio una prima classificazione della domanda può avvenire con i seguenti criteri:

- 1) se la domanda inizia o comprende il termine “Chi” ci si riferisce a una persona e quindi possiamo assegnare a una variabile QType il valore PERSON
- 2) Se la domanda inizia o comprende il termine “Dove”, QType indicherà un luogo assumendo il valore LOCATION.

In definitiva i nuovi tipi di sistema QA assumono la responsabilità di generare risposte soddisfacenti, piuttosto che produrre una lista di indici, potenzialmente irrilevanti e ottenuti secondo un ragionamento probabilistico. Dal momento che non sempre le risposte fornite possono essere esaurienti è naturale che un sistema QA, deve anche essere in grado di supportare ulteriori richieste dell’utente, e adattarsi al processo di ricerca con un atteggiamento collaborativo e *dialogante*.

Ma, prima di procedere oltre, che cosa rende il dialogo differente dagli altri sistemi di discorso? Ad esempio da una relazione, da un breve resoconto, da un monologo?

Possiamo dire che lo svolgimento di un dialogo, e il suo sviluppo coerente presenta delle caratteristiche analoghe ma notevolmente differenti da quelle delle forme di discorso a cui abbiamo accennato.

Interessante, nella teorizzazione di un chatbot, risulta il ragionamento di Leonard Bloomfield⁹⁶.

Bloomfield ipotizza la presenza di due eventuali parlanti, Jack e Jill insieme in una passeggiata lungo un breve viale. Jill è affamata e vede un albero di mele dietro un recinto. Produce quindi questa specie di “rumore” con la propria laringe, la lingua e le labbra. Jack riceve e interpreta il “rumore” supera il recinto si arrampica sull’albero, coglie una delle mele e la riporta indietro a Jill, ponendola sulla sua mano. Jill mangia la mela.

Questa successione di eventi può essere studiata in molteplici maniere ma nell’ambito di uno studio linguistico, potremo distinguere tra il momento, l’atto vero e proprio di espressione in un linguaggio, e gli altri eventi, i cosiddetti *practical events*.

⁹⁶ Bloomfield L., *Language*, p. 18, Henry Holt, New York (1933).

L'avvenimento considerato si compone in questo modo di tre parti, in ordine di tempo.

A: Practical events che precedono l'act of speech

B: Speech

C: Practical events che seguono l'act of speech

Bloomfield considera, innanzi tutto, la prima serie di *Practical events* e gli stimoli che spingono il parlante Jill a porre la propria richiesta nei confronti dell'altro parlante che si trova insieme a lei, Jack, disponibile ad ascoltare quest'ultima: tali eventi possono definirsi come stimoli del parlante.

Passando invece ad analizzare gli eventi che seguono l'atto linguistico (contrassegnati con C), essi producono il diretto coinvolgimento di Jack ma interessano in ogni caso Jill (che riceve la mela e la mangia).

Ovviamente questa sequenza di eventi è solo una delle possibili sequenze che possono riguardare la situazione descritta e che potrebbe addirittura svolgersi senza il ricorso ad alcuno *speech act*.

Ricorso che, per altro, è inteso da Bloomfield come una sorta di risorsa aggiuntiva a quelle già a disposizione di Jill per raggiungere lo scopo di mangiare una mela. Una delle conclusioni dello studioso è esattamente questa: il linguaggio permette a un individuo di compiere una reazione (R) quando un altro ha uno stimolo (S).

In una situazione ideale, all'interno di un gruppo di persone che possono comunicare attraverso il linguaggio, ognuno ha a disposizione le risorse e le abilità delle altre persone facenti parte del gruppo.

Bloomfield ritiene per altro opportuno distinguere come i tipi di reazione da parte dei vari individui a uno stimolo verbale possono essere essi stessi degli atti verbali, secondo lo schema:

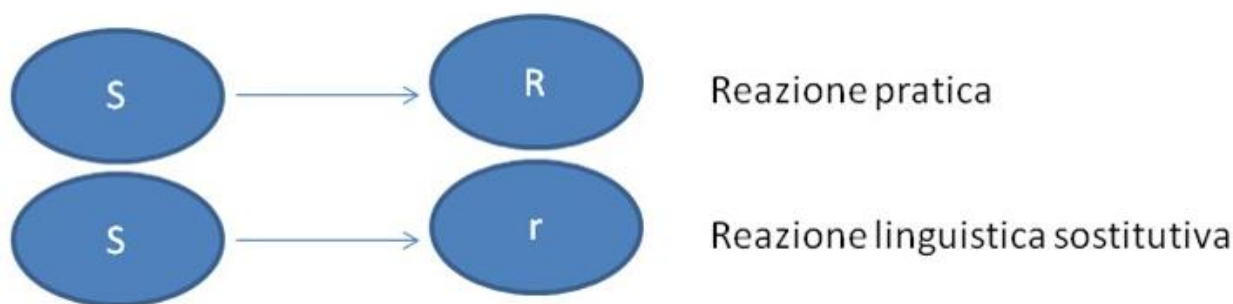


Fig. 45 - Reazione stimolo risposta

È proprio a questi due tipi di reazione che ci interessiamo nell'ambito del nostro studio: nell'alternanza stimolo e reazione il valore di "significato" di ciò che è detto viene a determinarsi nello sviluppo stesso di tale alternanza, e in come le varie frasi pronunciate si rapportano con la realtà di riferimento.⁹⁷

⁹⁷ "Accordingly, we say that speech-utterance, trivial and unimportant in itself, is importante because it has a meaning: the meaning consists of the important things with which the speech-utterance (B) is connected, namely the practical events (A and C)". L. Bloomfield, *Language and Linguistics*, p. 27.

Ritornando al nostro discorso possiamo chiarire in modo più dettagliato una delle caratteristiche del dialogo, che distingue quest'ultimo, da altre forme di comunicazione: l'alternanza di turno fra i vari partecipanti.

Il parlante A dice qualcosa, quindi il parlante B risponde, e così via secondo modalità analoghe anche in caso di presenza di un altro interlocutore. Un esempio può risultare il seguente frammento tratto da un dialogo concernente la pianificazione di un viaggio, un ambito (quello dei servizi di assistenza) di grande interesse nelle ricerche nel dialogo uomo-macchina.

A1: ... avrei bisogno di viaggiare a maggio.

B1: e in che giorno desidera viaggiare?

...

Esaminiamo innanzi tutto una particolare questione. Come riescono i due parlanti a definire il tempo giusto di attesa per il prosieguo del dialogo?

Da studi accurati riguardanti le tempistiche delle frasi pronunciate all'interno di una conversazione e delle pause tra tali espressioni, è possibile riscontrare che non esistono sovrapposizioni ragguardevoli tra i vari parlanti.

In poche parole l'inizio del turno di ogni parlante segue la fine del turno del parlante che lo ha preceduto. Nell'inglese americano la percentuale di sovrapposizioni, misurata da Steve Levinson⁹⁸ risulta inferiore al 5% e addirittura inferiore a questa percentuale per certi tipi di dialogo orientati verso uno scopo (come quello della prenotazione di un volo, richiesta dell'orario di un treno, ecc.).

Da questo presupposto, se i parlanti non si sovrappongono potrebbe sembrare che essi attendano un tempo prefissato dopo l'ultimo turno, eppure anche questo fenomeno risulta piuttosto raro. Infatti la quantità di tempo che trascorre tra i vari interventi è piuttosto piccola, generalmente dell'ordine di poche centinaia di millisecondi, anche in discorsi dove sono presenti più interlocutori.

Sembra addirittura che la pianificazione della successiva frase da pronunciare e la relativa attivazione delle procedure motorie per realizzarla si sviluppino, prima che il parlante in attività finisca il proprio turno.

Affinché ciò si verifichi, quindi, è necessario che una conversazione sia impostata secondo una particolare struttura che rende i parlanti (per la maggior parte dei casi) in grado di prevedere chi avrà il successivo turno per esprimersi, e quando ciò avrà luogo. Questo genere di comportamento, costruito sull'alternanza dei turni, è generalmente studiato nel campo dell'analisi della conversazione (Conversational Analysis - CA).

Secondo H. Sacks⁹⁹, tale comportamento è governato (almeno nell'inglese americano, ma il ragionamento è estendibile ad altre lingue) da un insieme di regole che permettono di prevedere il verificarsi dei momenti di alternanza. Queste regole determinano l'esistenza dei cosiddetti TRP (transition-relevance place), luoghi cioè dove la strutturazione stessa della lingua facilita l'alternanza degli interventi dei vari parlanti.

Alcuni dei TRP vengono spesso utilizzati in modo implicito da ogni parlante e sono sostanzialmente piuttosto ovvi come nel caso delle seguenti regole:

⁹⁸ Levinson Steve, *Pragmatics*, Cambridge University press, Cambridge (1983).

⁹⁹ Sacks H., Schegloff E. A., Jefferson G., "A simplest systematics for the organization of turn-taking for conversation", *Language*, 50(4), pp. 696-735.

a) se durante il proprio turno, il parlante A ha selezionato B come suo successore nella conversazione, allora B deve intervenire.

b) se il parlante corrente, A, non seleziona il prossimo parlante a intervenire, chiunque prende parte alla conversazione può farlo.

c) se nessuno interviene in seguito al compimento dell'intervento del parlante A, quest'ultimo può riprendere la parola.

Come già detto esse sono implicitamente applicate nelle conversazioni comuni ma definiscono in maniera *implicita* come il nostro dialogare segue delle vere e proprie convenzioni opportunamente strutturate. La regola a) codifica ad esempio lo schema QUESTION-ANSWER dove appunto il parlante A, seleziona il parlante B per rispondere a una domanda. L'accoppiata QUESTION-ANSWER viene definita come coppia adiacente da E. Schlegoff¹⁰⁰: una delle possibili coppie con queste caratteristiche è ad esempio la coppia GREETING-GREETING, nella quale si denota come a un saluto si risponda tendenzialmente con un altro saluto; la coppia COMPLIMENT-DOWNPLAYER, suggerisce invece che a un complimento si risponde con una attestazione di modestia oppure ringraziando semplicemente; funzionamento analogo ha la coppia di espressioni adiacenti secondo lo schema APOLOGY-MINIMIZATION (a delle scuse si risponde minimizzando il problema che le ha provocate), mentre la coppia BLAME - DENIAL / ADMISSION fa seguire a un rimprovero o biasimo, l'ammettere o il negare la colpa; infine possiamo ancora segnalare la coppia REQUEST-GRANT nella quale a una richiesta segue una risposta in grado di soddisfare o meno tale richiesta.



Fig. 46 - Realizzazione del semplice schema GREETING-GREETING

¹⁰⁰ Schegloff E. A., "Sequencing in conversational openings", *American Anthropologist*, 70, pp. 1075-1095 (1968). Il problema risulta riscontrabile anche all'interno della progettazione di un chatbot nel quale le modalità di input e output (scritta o orale) determinano in modo significativo l'impostazione del chatbot stesso.

Una coppia adiacente è definita come una struttura di due frasi che posseggono la caratteristica di essere adiacenti, prodotte da diversi parlanti, ordinate in una prima parte e in una seconda e definite in maniera tale che a una prima parte ne segue una seconda connessa a quest'ultima secondo un determinato schema



Fig. 47 - Schema APOLOGY- MINIMIZING

All'interno della regola a) si può configurare quello che viene definito come *significant silence*, o silenzio provvisto di significato, nel quale la pausa successiva alla richiesta di un intervento, se eccessiva, deve essere interpretata come una pausa conferente significato al dialogo.

Ad esempio:

- A: Is there something bothering you or not?
(1.0)
- A: Yes or no?
(1.5)
- A: Eh?
- B: No.

Il silenzio manifesto in due successive richieste viene interpretato come rifiuto di rispondere oppure addirittura come risposta negativa.

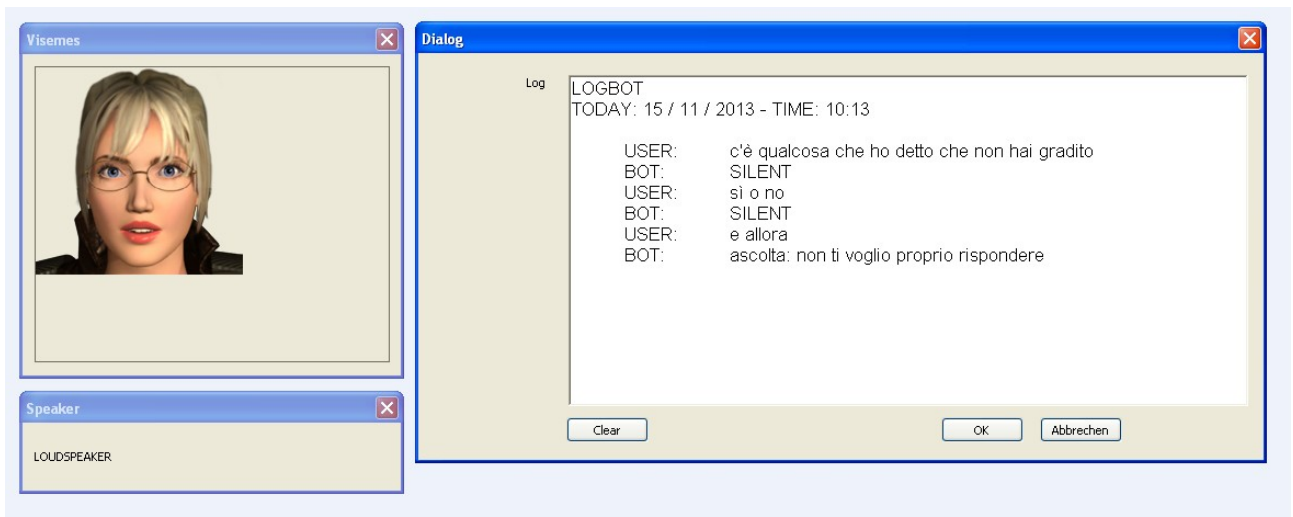


Fig. 48 - Realizzazione nel chatbot.

Ritornando al nostro discorso sui TRP, ribadiamo come essi di solito non si verifichi ovunque nell'equilibrio di una conversazione, bensì si distribuisca generalmente ai confini dei vari enunciati. Questa peculiarità definisce un'altra delle differenze tra enunciato scritto, dove tali confini sono espressi graficamente, e parlato, dove subentrano fattori diversificati, in grado di strutturare differentemente la distribuzione della conversazione.¹⁰¹

Essenzialmente gli enunciati scritti differiscono da quelli parlati in una serie di caratteristiche peculiari: tendono a essere più lunghi, vengono utilizzate frasi nominali in sostituzione dei semplici pronomi, esiste una gestione diversa delle pause, delle correzioni dovute a fraintendimenti, delle ripetizioni.

Ma soprattutto, ciò che risulta oltremodo evidente (come abbiamo accennato) è la maggiore difficoltà nel segmentare le espressioni orali.

Le frasi scritte e i vari paragrafi della scrittura possono essere segmentati in maniera automatica, mentre nell'ambito degli enunciati orali tale segmentazione presenta una maggiore complessità. Individuare i confini dei vari enunciati è importante poiché molti modelli di interfacce dialoganti sono basati sulla necessità di estrarre un enunciato come una unità primitiva. Nel caso di una interfaccia dialogante con due agenti il problema può essere risolto con una artificiosa gestione dei *timer*¹⁰² nell'applicazione, ma ovviamente nel caso di più agenti, la complessità cresce in modo esponenziale. In ogni caso il problema della segmentazione è di difficile gestione poiché un singolo enunciato può svilupparsi all'interno di uno o più turni, o anche un singolo turno può includere diversi enunciati. Per esempio nel seguente frammento di dialogo tra un agente di viaggio e un cliente, l'enunciato espresso dall'agente coinvolge tre turni di dialogo:

- A: Yeah yeah the um let me see here we've got you on American flight nine thirty eight.
B: Yep.
A: Leaving on the twentieth of June out of Orange County John Wayne Airport at seven thirty p.m.
B: Seven thirty.
A: And into uh San Francisco at eight fifty seven.

Al contrario l'esempio seguente presenta tre enunciati all'interno di un solo turno:

- A: Three two three and seven five one. OK and then does he know there is a nonstop that goes from Dulles to San Francisco? Instead of connection through St. Louis.

Gli algoritmi tradizionali per la gestione della segmentazione degli enunciati sono basati su numerosi criteri.

Ad esempio:

1) *cue words*. Parole chiave come “bene”, “perciò”, “cioè” tendono a presentarsi all'inizio o alla fine degli enunciati.

¹⁰¹ È generalmente risaputa l'impossibilità di trasferire in forma scritta un dialogo parlato, nell'ambito della narrativa: semplici trascrizioni sembrerebbero non funzionare allo scopo di conferire al dialogo scritto quella vivacità, quel ritmo e quel timbro che un *lettore* si aspetterebbe.

¹⁰² I *Timer* sono componenti in grado di gestire il *multithreading* ossia la gestione di più istruzioni e sequenze di esse, in apparente contemporaneità. Nella gestione del mio chatbot ho utilizzato molteplici timer per simulare diversi flussi di pensiero e diverse istanze elaborative delle domande poste e degli argomenti trattati.

2) *n-gram word sequences*. Una specifica sequenza di parole indica spesso un punto di confine. Le grammatiche basate sugli *N-Grams*¹⁰³ possono essere modellate e annotate con i tag indicanti tali sequenze e successivamente rielaborate per individuare ulteriori sequenze con queste caratteristiche.

3) caratteri prosodici: accenti, alterazioni del tono della voce possono essere sfruttati analogamente.

Grounding

Come abbiamo abbondantemente ricordato un'altra importante caratteristica che distingue un dialogo da un monologo consiste nel fatto che un dialogo risulta un atto collettivo realizzato da un parlante e un ascoltatore. Un'indicazione di questa *collettività* si evidenzia quando, a differenza da un monologo, il parlante e l'ascoltatore devono costantemente stabilire un comune contesto argomentativo, o comunque un terreno comune su cui sviluppare i propri enunciati: il cosiddetto *common ground*.¹⁰⁴

Il bisogno di ottenere questi presupposti indica che l'ascoltatore deve essere in grado di riconoscere e in un certo senso comprendere gli enunciati del parlante, o altrimenti manifestare con chiarezza problemi esistenti per un successivo sviluppo della conversazione.

Ad esempio, si consideri il ruolo della interiezione "mm hmm" nel seguente frammento di dialogo tra un agente di viaggio e il suo cliente:

A: ...returning on US flight one one one eight.
B: Mm hmm.

L'espressione "mm hmm" qui risulta fungere da continuatore, ed è comunemente chiamata *backchannel* o *acknowledgement token*.

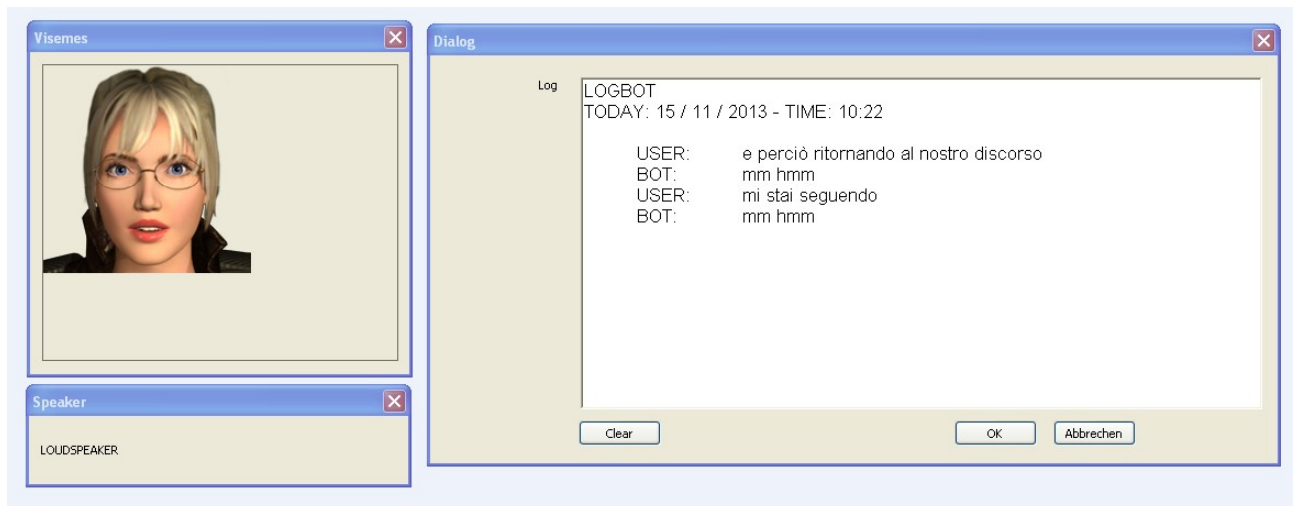


Fig. 49 - Espressioni di acknowledgement

¹⁰³ "N" parole consecutive.

¹⁰⁴ Stalnacker R. C., *Assertion*, in Cole P., *Pragmatics: Syntax and Semantics, Volume 9*, pp. 315-332, Academic Press, New York (1978).

Un *continuatore* risulta in definitiva un breve enunciato che riconosce, in qualche modo, quello precedente e sollecita il parlante nella prosecuzione del suo discorso. Rendendo noto al parlante che l'ascoltatore ha ricevuto e compreso l'enunciato, un *continuatore* aiuta il parlante e lo stesso ascoltatore a sintonizzare il proseguimento della conversazione su un terreno comune.

Operazioni inverse a quelle dei continuatori possono essere svolte da espressioni di floor-holding (che consentono al parlante, di continuare a parlare) e di floor-grabbing (che invitano l'interlocutore a tacere affinché il nuovo parlante subentri nel turno del dialogo).

Ad ogni modo, ovviamente, i continuatori sono solo una delle molteplici maniere attraverso le quali l'ascoltatore indica al parlante di credere di aver capito quello che il parlante vuole intendere.

H. Clark e E. Schaefer¹⁰⁵ individuano cinque principali metodi attraverso i quali è possibile svolgere tale funzione:

a) attenzione continuata: il parlante B mostra di continuare ad attendere un ulteriore sviluppo della conversazione e perciò è soddisfatto dalla presentazione effettuata dal parlante A.

A: Questa è l'auto più veloce in circolazione.
B: È anche una delle più costose.

b) prossimo contributo rilevante: il parlante B introduce il successivo argomento rilevante nel filo del discorso.

A: Conosce il celebre attore Roberto Rossi.
B: Certo ho visto il suo ultimo film.

c) riconoscimento: B annuisce o si esprime con un continuatore, una interiezione come "aha", "già" o un'espressione come "ben detto", "proprio così".

A: Penso che nel nostro paese dovrebbero diminuire le tasse.
B: Ben detto. La penso allo stesso modo.

d) dimostrazione: B dimostra che ha compreso tutto o una parte di quanto espresso da A., per esempio parafrasando o riformulando ciò che A ha detto oppure collaborando a completare il suo enunciato.

(si consideri la figura 50)

e) esposizione (*display*): B spiega a parole tutta o parte dell'esposizione di A

Ad esempio il seguente campione di conversazione mostra una forma di *display* della comprensione attraverso la ripetizione del finale di frase del parlante A

A: D'accordo, prenderò il primo volo che parte in giornata.
B: Il primo volo che parte in giornata?

¹⁰⁵ Clark H., Schaefer E. F., "Contributing to discourse". *Cognitive Science*, 13, pp. 259-294 (1989).

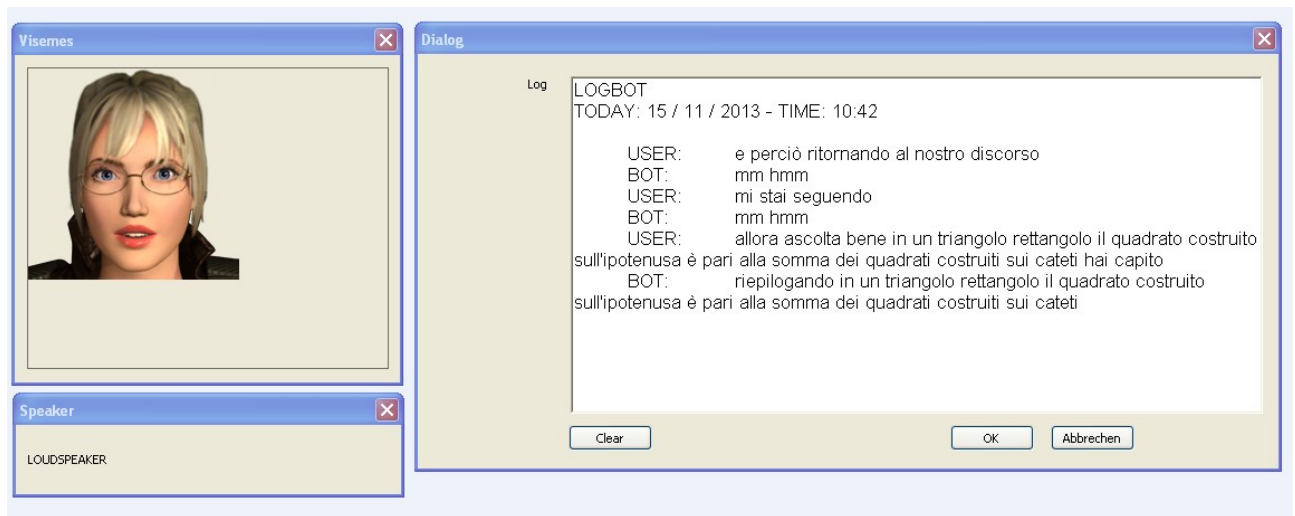


Fig. 50 - Esempio di *Dimostrazione* nel chatbot

Ovviamente tutte queste espressioni possono essere sfruttate, come evidenziato, nella progettazione di un chatbot proprio allo scopo di segmentare le diverse fasi del dialogo e aggiungere significato a queste ultime. Ciò nonostante tale gestione è strettamente collegata alla qualità della *speech engine*, sia a livello di riconoscimento vocale che di sintesi testuale.

Inoltre come nel caso della regola (e) è piuttosto semplice realizzare un algoritmo in grado di mantenere l'argomentazione in "sintonia" tra entrambi gli interlocutori. Alcune dei metodi di Clark e Shafer possono per altro essere applicati soltanto con interfacce visuali che permettono, come nel caso dell'attenzione continuata, di gestire aspetti visivi (ad esempio gli indicatori di attenzione forniti da alcune espressioni facciali). Tali procedimenti sono in ogni caso di notevole importanza poiché è stato rilevato come gli utenti di interfacce dialoganti vedono proprio in questa assenza di espliciti segnali di riconoscimento, una delle carenze più significative dei sistemi stessi.

Esistono in ogni caso molteplici altri atti o espressioni di riconoscimento analoghi a quelli considerati. Si pensi alle richieste di correzione o ripetizione, attuabili in molteplici modalità ad esempio anche attraverso l'inserimento di una interiezione

- A: Why is that?
- B: Huh?
- A: Why is that?

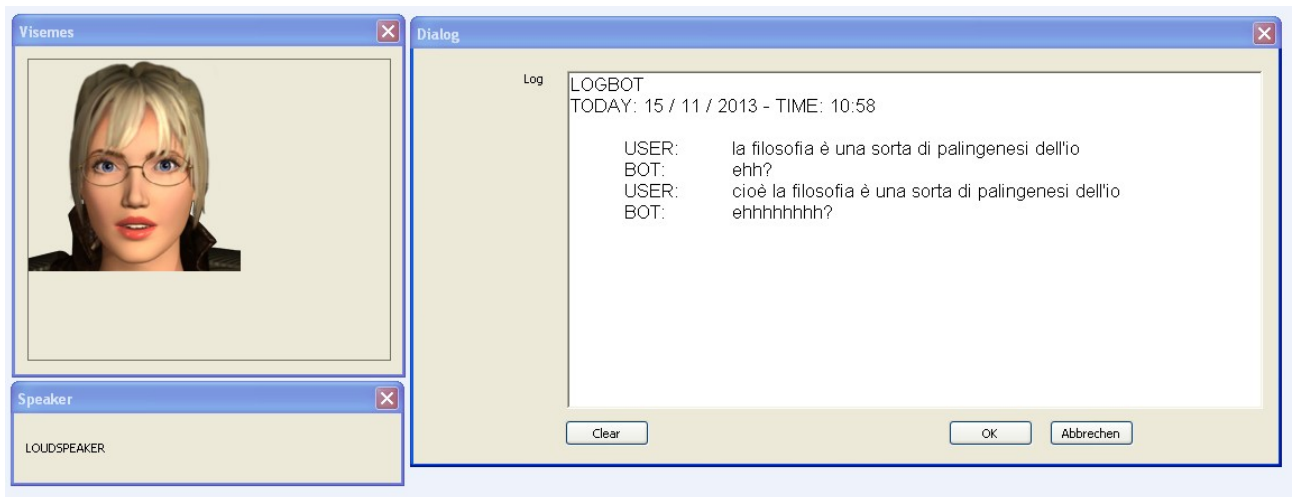


Fig. 51 - Esempio di interiezioni

Implicazione conversazionale

Una delle più importanti proprietà di un dialogo è la maniera con la quale un enunciato fa riferimento a un significato diverso da quello letterale. Consideriamo la risposta del cliente B nell'esempio di una delle conversazioni precedenti:

- A: And, what day in May did you want to travel?
B: OK uh I need to be there for a meeting that's from the 12th to the 15th.

In questo caso, come possiamo osservare, il cliente non risponde semplicemente alla domanda. Il cliente asserisce in sostanza che ha un appuntamento a un determinato orario. Una interpretazione semantica di tale frase sarà incentrata proprio sull'appuntamento in questione.

Esempio di realizzazione nel Bot.

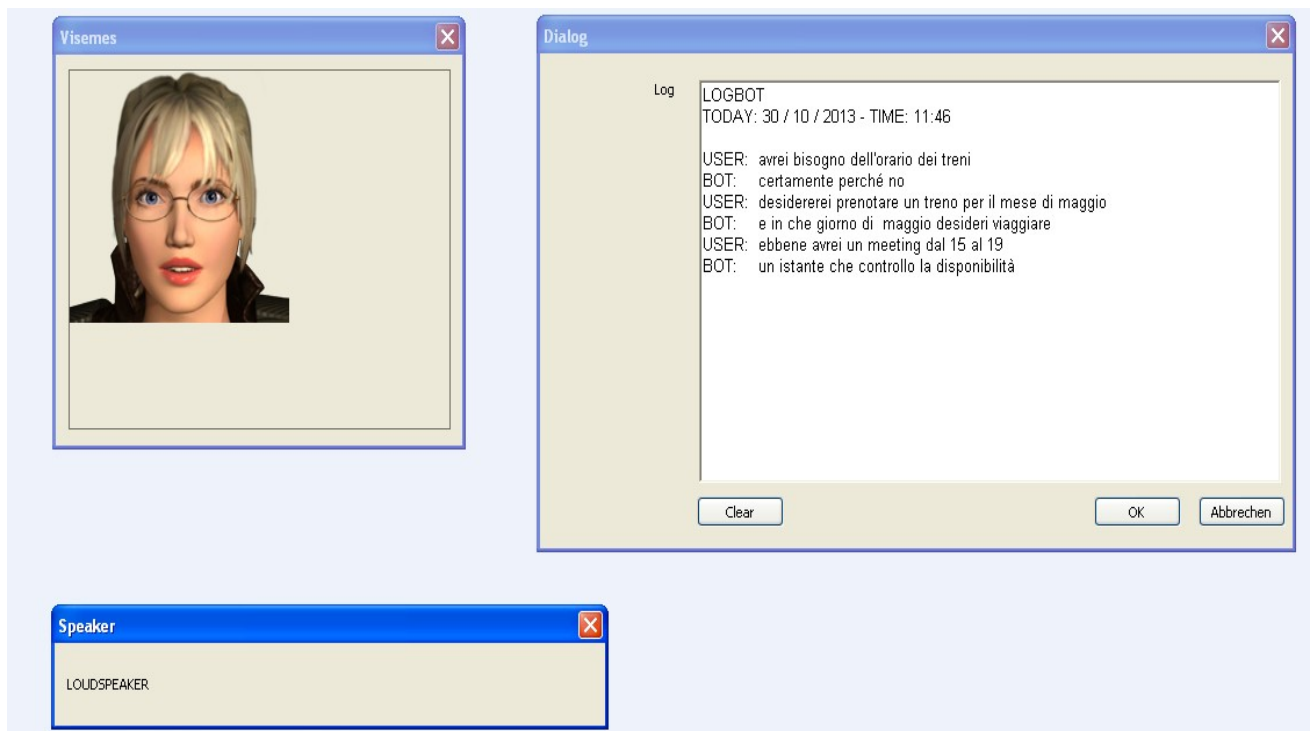


Fig. 52

Cosa induce l'agente del sistema dialogico a ritenere che il cliente ha menzionato il meeting soltanto per dare informazioni sulle date del suo viaggio?

Consideriamo un altro campione dello stesso dialogo. Questa volta l'agente stesso dice:

A: ...There's three non-stops today.

Questa asserzione potrebbe essere considerata vera, anche se ci fossero sette corse senza fermata quest'oggi, poiché in questo caso ce ne sarebbero automaticamente anche tre. Ma ciò che il parlante qui vuol dire, oltre al valore effettivo dell'asserzione, è che ci sono tre corse con quelle caratteristiche e non più di tre corse. Come riesce l'ascoltatore a dedurre ciò che il parlante, in questo caso l'agente di viaggio, lascia sottinteso?

I due casi appena presentati, naturalmente, hanno delle caratteristiche in comune. In entrambi, il parlante sembra aspettarsi che l'ascoltatore sia in grado di effettuare determinate deduzioni; in altre parole il parlante sta comunicando ulteriori "dati" rispetto a quelli apparentemente presenti nelle parole pronunciate. Tali tipi di esempio furono osservati e studiati da H. P. Grice¹⁰⁶, nella sua teoria delle implicazioni conversazionali.

Una implicazione indica una particolare classe di prestabilite inferenze. Secondo Grice ciò che abilita gli ascoltatori a effettuare questo tipo di deduzioni, è insito in determinate caratteristiche, proprie di ogni conversazione, poiché ogni conversazione è guidata da un

¹⁰⁶ Grice H. P., "Logic and Conversation", *Speech Acts: Syntax and Semantics Volume 3*, pp. 41-58. Academic Press, New York (1975).

insieme di massime, dei procedimenti *euristici*¹⁰⁷ a carattere generale, che svolgono un ruolo chiave nella interpretazione degli enunciati conversazionali.

Essenzialmente Grice¹⁰⁸ propose le seguenti quattro massime:

1) massima della quantità: “non essere reticente o ridondante”. Bisogna dare tutte le informazioni esatte che sono richieste. Non bisogna contribuire con informazioni aggiuntive a quelle che sono effettivamente postulate.

2) massima della qualità: "sii sincero, fornisci informazione veritiera, secondo quanto sai". È necessario che il contributo conversazionale del parlante sia vero. Non bisogna dire ciò che si crede falso, non bisogna dire ciò per cui non si dispone di prove sufficienti atte a dimostrarne la veridicità.

3) massima della pertinenza: “sii pertinente”. È necessario essere pertinenti, nel puro senso della parola. Ciò che si dice deve avere una sua importanza condivisa tra il parlante e i suoi ascoltatori.

4) massima della maniera: “evita l’ambiguità”. È necessario essere chiari; evitare espressioni oscure, evitare ambiguità, essere brevi, evitare di essere prodighi di parole laddove non è necessario, essere ordinati.

Riferendoci ad esempio al dialogo tra l’agente di viaggio e il suo cliente, è proprio l’applicazione della massima della quantità, che permette all’ascoltatore di conoscere come l’accenno alle tre corse significhi che non ne esistono ulteriori.

Ciò accade poiché il parlante, qualora intendesse tale evenienza, lo specificherebbe direttamente.

La massima della rilevanza, nella stessa conversazione, permette all’agente di conoscere che il cliente vuole viaggiare entro e non oltre il 12 del mese di maggio: “I need to be there for a meeting that is from the 12th to the 15th”. L’agente assume che il cliente stia seguendo tali massime, e da qui il suo semplice menzionare l’appuntamento futuro, implica ciò che in realtà ha intenzione di significare. Queste tre proprietà della conversazione (*turn taking*, *grounding* e *implicature*) svolgeranno un ruolo importante nella descrizione degli atti dialogici, della struttura di un dialogo e nella gestione del dialogo stesso.

Atti dialogici.

Importanti considerazioni riguardanti le modalità di svolgimento di un dialogo si devono anche agli studi di John Langshaw Austin,¹⁰⁹ per il quale all’interno di un dialogo è insito un particolare tipo di azione effettuato in modo implicito dal parlante. Ciò è particolarmente comprensibile in frasi *performative*¹¹⁰ come le seguenti:

¹⁰⁷ Nell’ambito informatico, l’euristica è una tecnica progettata per risolvere un problema molto velocemente, al contrario di quanto occorrerebbe utilizzando i metodi classici. Tramite ottimizzazione, completezza, cura e precisione si ottengono soluzioni ottimali a un problema, l’euristica invece è in grado di produrre soluzioni, seppure non ottimali, in modo rapido che risolvono comunque il problema a portata di mano.

¹⁰⁸ Grice Paul H., *Logic and conversation in Syntax and semantics 3: Speech acts*, pp. 41-58, a cura di P. Cole, Academic Press, New York (1975).

¹⁰⁹ Austin J. L., *How to Do Things with Words*, Harvard University Press, Cambridge (1962).

¹¹⁰ L’atto *performativo* fa parte della teoria degli atti linguistici elaborata da Austin. Vi sono asserzioni che non descrivono un certo stato delle cose, non espongono un qualche fatto, in modo vero o falso. Mentre si enuncia quella frase si fa qualcosa: mediante l’atto *performativo* si compie quello che si dice di fare, conseguentemente si produce immediatamente un fatto reale. Il termine deriva, ovviamente, dall’inglese *perform*, eseguire.

- 1) battezzo questa nave Titanic
- 2) accetto la mozione
- 3) scommetto dieci dollari che domani neviccherà.

Quando tali frasi sono pronunciate, dalla giusta “autorità parlante”, esse producono un determinato, specifico effetto. Per esempio la prima frase produce l’effetto di cambiare la configurazione dell’ambiente circostante (facendo sì che una nave, qualora presente, prenda il nome *Titanic*), proprio allo stesso modo con cui una qualsiasi azione può cambiare lo stato in cui si trova il nostro ambiente di riferimento. Verbi come battezzare o approvare, che svolgono proprio questo tipo di azioni sono chiamati verbi performativi, e Austin chiamò tali azioni *speech acts*, o atti che possono essere eseguiti attraverso le parole. Ciò che rende il lavoro di Austin così rilevante è che gli *speech acts* non sono confinati a una ristretta classe di verbi performativi. Austin fa osservare che qualsiasi enunciato in una frase, all’interno di una situazione di dialogo, può svolgere tre tipi di atti:

- 1) atto locutorio: che indica l’espressione di una frase con un particolare significato.
- 2) atto illocutorio: cioè l’atto di chiedere, rispondere, promettere, effettuato nel momento in cui si pronuncia una frase.
- 3) atto perlocutorio: tale atto indica la capacità di produrre, spesso intenzionalmente, determinati effetti nei confronti dei sentimenti, dei pensieri, o delle stesse azioni di chi sta ascoltando la frase in questione.

Per esempio, Austin spiega che la frase “non puoi farlo” ha una implicazione illocutoria nell’interrompere l’attività di un ascoltatore, disturbando e nel caso interrompendo quest’ultimo.

La notazione “*speech act*” è generalmente usata per descrivere atti illocutori, piuttosto che gli altri due tipi di atti. J. R. Searle¹¹¹ proponendo una analoga classificazione suggerisce che tutti gli *speech act* possono essere classificati in 5 principali categorie:

- 1) assertivi o rappresentativi: atti come suggerire, giurare, concludere, nei quali il parlante pone l’accento sulla verità e sulla validità di ciò che dice.
- 2) direttivi: nei quali il parlante cerca di convincere l’ascoltatore a fare qualcosa (chiedere, ordinare, invitare...)
- 3) commissivi: il parlante si impegna a effettuare determinate azioni nel futuro (promettere, pianificare, scommettere, opporsi)
- 4) espressivi: esprimono lo stato, la condizione psicologica del parlante riguardo una determinata situazione (si effettuano nel ringraziare, nello scusarsi, nel dare il benvenuto, deplorare)
- 5) dichiarativi: determinano l’affermarsi di un diverso “stato del mondo” attraverso l’espressione dell’enunciato (includono molti degli atti performativi già menzionati: ad esempio: “ rassegno le mie dimissioni”, “tu sei licenziato”).

¹¹¹ Searle J. R., *A taxonomy of illocutionary acts*, in Gunderson K, *Language, Mind and Knowledge*, Minnesota Studies in the Philosophy of Sciences, Vol. VII, pp. 344-269, University of Minnesota Press, Amsterdam (1975).

Mentre gli *speech act* permettono una utile caratterizzazione di determinati aspetti pragmatici del linguaggio, recenti studi, specialmente nella costruzione di sistemi dialogici, hanno amplificato in modo significativo questa nozione, modellando una maggiore quantità delle funzioni conversazionali che una determinata frase può esprimere. Gli atti risultanti da questa nuova caratterizzazione sono chiamati nella notazione inglese *conversational moves*¹¹².

In ogni caso, associare a una espressione un determinato *speech act* presenta sovente delle difficoltà a causa della cosiddetta *indirect force*: le frasi possono avere, ad esempio, un *forza illocutoria* e al tempo stesso una ulteriore estensione che rappresenta la vera funzione dominante della frase stessa. Per esempio la frase “*Puoi passarmi il sale?*” si presenta nella forma di una domanda, ma la sua *forza indiretta* è la richiesta di passare il sale: la *forza indiretta* prevale sulla *forza letteraria*.

Altri tentativi di sviluppare una classificazione schematica di queste funzioni o *speech acts* è offerta dall’architettura DAMSL (Dialogue Act Markup in Several Layers)¹¹³.

In riferimento alla possibilità di tali atti, un modulo del mio sistema dialogico è costruito proprio nella definizione di un parametro “action”, gestito anch’esso da un timer che si occupa di quegli atti linguistici precedentemente annotati nelle strutture di markup costituenti le risposte del chatbot:

Si faccia riferimento alla seguente richiesta dell’utente.

User: Vorrei scrivere qualcosa.

La risposta è gestibile a partire da una struttura della base di conoscenza come la seguente

```
<category> <pattern><c> * scrivere qualcosa </c></pattern>
  <template><file>certamente</file><action>scrivere</action></template>
</category>
```

A una variabile corrispondente al parametro “action” viene assegnato il valore “scrivere” che induce il chatbot ad avviare il programma di videoscrittura predefinito.

¹¹² Power R., “The organization of purposeful dialogs”, *Linguistics*, 17, pp. 105-152 (1979).

¹¹³ Una copia delle notazioni del DAMSL può essere consultata all’indirizzo:

<http://www.cs.rochester.edu/research/speech/damsl/RevisedManual/>

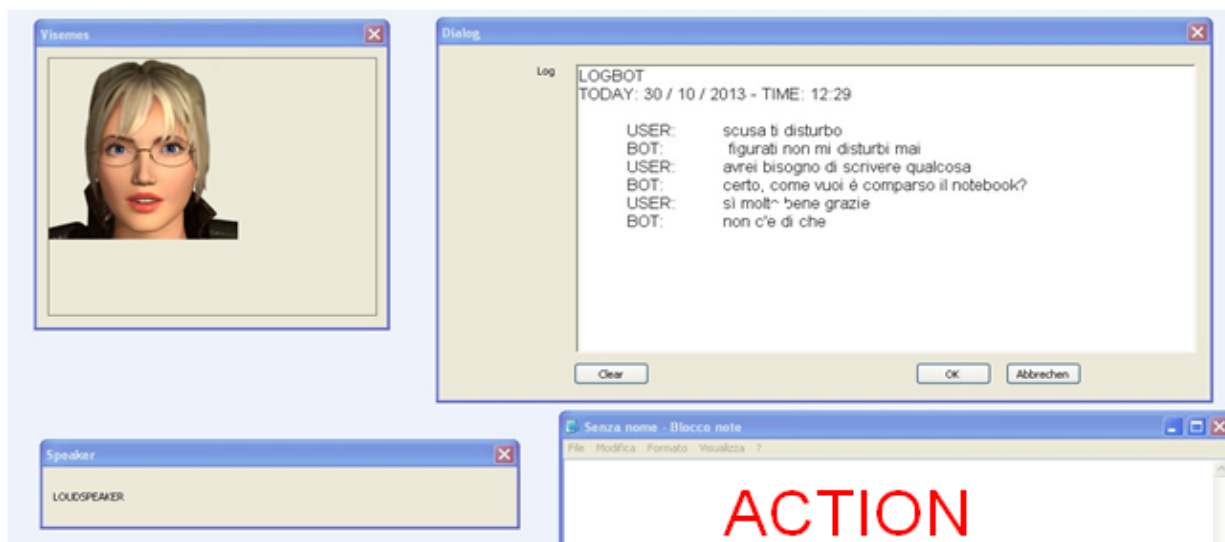


Fig. 53 - Esempio di richiesta dell'azione "scrivere".
 La richiesta è stata esaudita: si noti la comparsa del notebook sullo sfondo.

Il DAMS risulta un sistema per annotare i dialoghi: definisce e classifica le varie frasi, il loro ruolo all'interno del dialogo e le reciproche relazioni. Può essere adattato alle più svariate esigenze e ai più svariati campi di applicazione.

Una annotazione di esempio è la seguente:

```

T1 utt1: s: uh would take two hours <sil> [ assuming ](1) you have
| |           an engine at Bath
T2 utt2: u: [ okay ](1)
  
```

T1 e *T2* indicano i turni del dialogo, *utt1*, *utt2*, definiscono le due *utterance* dell'annotazione, *s* ed *u* sono i due parlanti e in questa annotazione specifica i termini tra parentesi quadra sono contrassegnati dal numero 1 che indica il sovrapporsi della loro pronuncia nel corso del dialogo.

Nella seguente annotazione invece le due parentesi quadre giustappunte segnalano una pausa del parlante *s*: la pausa viene riempita dal parlante *u* e quindi ciò che viene detto da quest'ultimo ("Avon") confermato.

```

T1 utt1: s: so you've got the engines at Elmira and uh [](1) Avon
T2 utt2: u: [Avon](1)
  
```

Nel sistema DAMS esistono quattro categorie di annotazioni. Nella prima categoria si segnala il cosiddetto *Communicative-Status* che definisce se la frase è non interpretabile, interrotta, o se il parlante sta rivolgendosi a se stesso indipendentemente dalla presenza dell'interlocutore.

Nella seconda categoria *Information-Level* le annotazioni forniscono una caratterizzazione astratta della frase. Nei dialoghi *task-oriented* si possono grosso modo dividere le frasi in quelle orientate al compimento del task, in quelle che hanno uno scopo prettamente comunicativo (*Communication-management*), e in quelle che non rientrano in nessuna di queste categorie. Inoltre è possibile una ulteriore suddivisione tra le frasi che

determinano progressi nello stato di avanzamento del task e quelle che ne discutono semplicemente (*Task-management*)

Una ulteriore categoria dell'architettura DAMSL è costituita dalle cosiddette *backward looking functions* che indicano in quale modo una frase è in relazione con quelle precedenti e sono individuabili nel seguente schema:

AGREEMENT	risposta del parlante a una proposta precedente
ACCEPT	la proposta non viene accettata
ACCEPT-PART	una parte della proposta viene accettata
MAYBE	la proposta non viene né accettata né respinta
REJECT-PART	la proposta viene respinta in una sua parte
REJECT	la proposta viene respinta
HOLD	si pospone una eventuale risposta
ANSWER	risposta a una domanda
UNDERSTANDING	si chiede se il parlante ha compreso la domanda
SIGNAL-NON-UND	il parlante non ha capito
SIGNAL-UND	il parlante ha capito
ACK	la comprensione viene mostrata tramite un continuatore o un'altra asserzione
REPEAT-REPHRASE	la comprensione è dimostrata tramite ripetizione o riformulazione
COMPLETION	la comprensione viene dimostrata tramite un completamento del discorso

Tab. 10 - Classificazione di alcuni dei *Backward Looking Tags*

Il seguente schema indica invece le *forward looking functions* che invece producono e prevedono una reazione futura nell'interlocutore

STATEMENT	una assunzione prodotta dal parlante
INFO-REQUEST	una domanda del parlante
CHECK	domanda per confermare l'informazione
INFLUENCE-ON-ADDRESSEE	influenza il proseguimento del discorso (direttive di Searle)
OPEN-OPTION	un suggerimento con una lista di opzioni
ACTION-DIRECTIVE	un ordine
COMMITTING-SPEAKER	(da Austin)
OFFER	il parlante si offre di fare qualcosa
COMMIT	il parlante è praticamente costretto o si impegna a fare qualcosa
CONVENTIONAL	
OPENING	saluti
CLOSING	congedi
THANKING	ringraziamenti e risposte a ringraziamenti

Tab. 11 - Classificazione di alcuni dei *Forward Looking Tags*

Nel tentativo di individuare l'effettiva consistenza di tali atti dialogici ho cercato di realizzarli all'interno della mia interfaccia dialogante.

Ad esempio, sempre facendo ricorso al markup <action> posso descrivere le categorie per "ringraziare": THANKING.

```
<category>
<pattern><filepattern>thanking</filepattern></pattern>
<template><filetemplate>please</filetemplate><action>downstatement</action></
template>
</category>
```

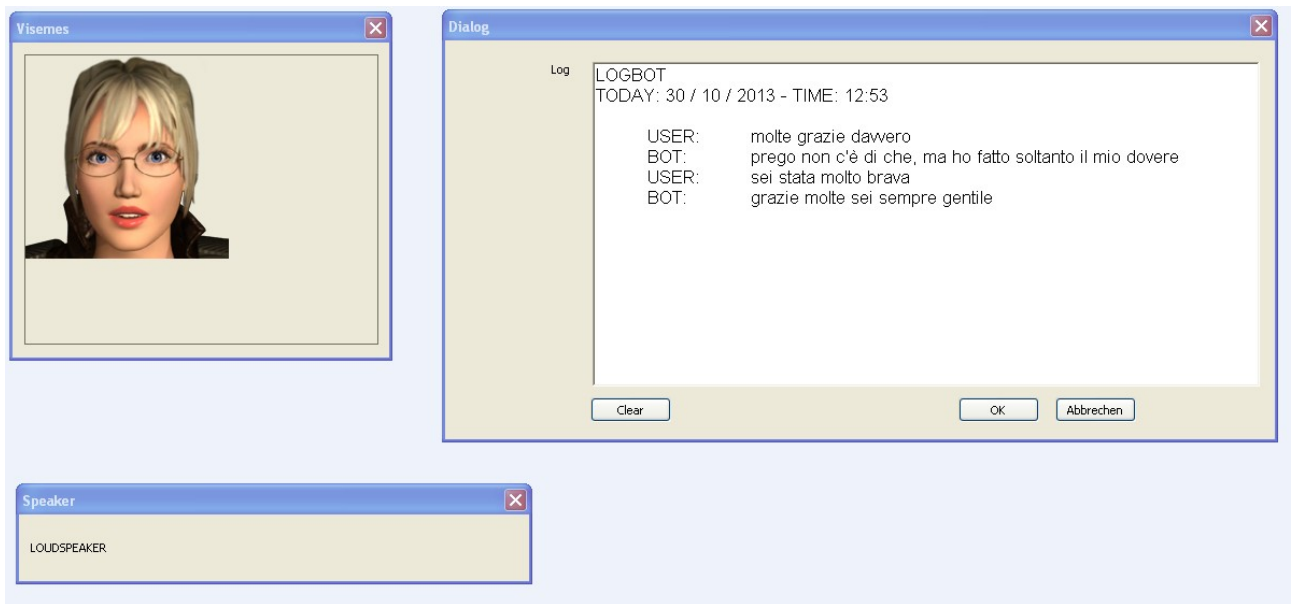



Fig. 54 - Esempio di downstatement gestito dal chatbot LUCY

Interpretazione automatica di atti dialogici.

Abbiamo appena introdotto i cosiddetti atti dialogici o le varie attività che un enunciato può produrre.

In questa sezione parleremo più specificatamente del problema di identificare e interpretare tali atti. In poche parole, come decidiamo se l'input proposto da un parlante sia una QUESTION, uno STATEMENT, un atto SUGGEST, o un ACKNOWLEDGEMENT?

In prima analisi tale problema risulterebbe semplice, anche se con caratteristiche diverse in dipendenza della lingua esaminata. Ad esempio in inglese le *yes-no-questions* sono anticipate dalla *aux-inversion*, gli statement hanno una impostazione dichiarativa (nessuna *aux-inversion*) e gli "ordini" hanno la sintassi delle frasi con i verbi all'imperativo.

Alcuni esempi di tali frasi:

YES-NO-QUESTION	Will breakfast be served on USAir 1557?
STATEMENT	I don't care about lunch
COMMAND	Show me flights from Milwaukee to Orlando on Thursday night

È possibile riscontrare come all'interno di una interfaccia dialogante, empiricamente, la distinzione tra lo statement e la *yes-no-question* non è così marcata poiché il silenzio successivo allo statement induce in ogni caso, come abbiamo visto proprio riguardo i cosiddetti TPF l'interlocutore a intervenire nel dialogo come se l'asserzione precedente sia a tutti gli effetti una domanda.

In ogni caso distinguere un atto illocutorio presenta determinate difficoltà.

Una possibile annotazione dei vari atti secondo la classificazione DAMSL della conversazione sulla quale abbiamo fatto riferimento è la seguente:

[assert]	C1: ... I need to travel in May.
[info-req, ack]	A1: And, what day in May did you want to travel?
[assert, answer]	C2: OK uh I need to be there for a meeting that's from the 12th to the 15th.
[info-req, ack]	A2: And you're flying into what city?
[assert, answer]	C3: Seattle.
[info-req, ack]	A3: And what time would you like to leave Pittsburgh?

[check, hold]	C4:	Uh hmm I don't think there's many options for non-stop.
[accept, ack]	A4:	Right.
[assert]		There's three non-stops today.
[info-req]	C5:	What are they?
[assert, open-option]	A5:	The first one departs PGH at 10:00 am arrives Seattle at 12:05 their time. The second flight departs PGH at 5:55pm, arrives Seattle at 8 pm. And the last flight departs PGH at 8:15 pm arrives Seattle at 10:28pm.
[accept, ack]	C6:	OK I'll take the 5ish flight on the night before on the 11th.
[check, ack]	A6:	On the 11th?
[assert, ack]		OK. Departing at 5:55pm arrives Seattle at 8pm, US Air flight 115.
[ack]	C7:	OK.

Tra i vari casi in cui la classificazione esposta, riguardante i vari atti, risulta di difficile analisi possiamo presentare il seguente. La frase:

A: Can you give me a list of the flights from Atlanta to Boston?

potrebbe essere intesa come una YES-NO-QUESTION e traducibile come *Are you capable of giving me a list of* oppure come una vera e propria richiesta dell'elenco di tali voli.

Va per altro evidenziato come in entrambi i casi le frasi proposte a un ipotetico chatbot indurrebbero quest'ultimo a svolgere un'azione. In LUCY la realizzazione di una analoga richiesta è stata la seguente:

```
<category>
  <pattern><c> Can you give me a list of the flights from * to
*</c></pattern></template>
  <filetemplate>certamente</filetemplate><action>listofflights</action></templa
te>
</category>
```

Nel modulo che gestisce tale richiesta le due località vengono memorizzate in una lista di variabili stringa e sono poi elaborate secondo le prerogative del modulo indicizzato dall'*action* "listofflights" che farà riferimento a un database di orari predefinito, eseguendo una ricerca a partire dai valori delle due variabili definenti l'orario.

La risposta immediata alla domanda, sarà invece resa da una delle risposte elencate nel file "certamente", che consiste, come vedremo più in dettaglio, in una lista di frasi "equivalenti" in grado di esprimere una risposta affermativa.

Ritornando al nostro discorso, un'altra delle categorie piuttosto comuni e semplici da individuare sono le cosiddette CHECK questions,¹¹⁴ quelle domande utilizzate per chiedere a un interlocutore di confermare il suo essere in possesso di una conoscenza precedentemente acquisita. Alcune di queste domande possono, per altro, avere una forma dichiarativa come evidenziato dall'asserzione posta in grassetto nel seguente dialogo, sempre tra un agente di viaggio e il suo interlocutore:

¹¹⁴ Carletta J., "The reliability of a dialogue structure coding scheme", *Computational Linguistics*, 23 (1), pp. 13-32 (1997).

A: OPEN-OPTION I was wanting to make some arrangements for a trip that I'm going to be taking uh to LA uh beginning of the week after week

B: HOLD OK uh let me pull up your profile and I'll be right with you here [pause]

B: CHECK **And you said you wanted to travel next week?**

A: ACCEPT Uh yes.

Tali enunciati, che usano la struttura superficiale tipica dell'*asserzione* per esprimere una domanda sono chiamati *indirect speech acts*.

Lo svilupparsi delle varie asserzioni, da una forma all'altra, si svolge attraverso un procedimento di inferenza che da un enunciato ne produce un altro, secondo regole di trasformazione che potrebbero essere codificate in modo deterministico a seconda della lingua di riferimento.

A questo proposito indicheremo di seguito due metodi per realizzare tale procedimento di inferenza.

Plan-Inferential Interpretation of Dialogues Acts

Questo approccio all'interpretazione degli atti di dialogo fu inizialmente proposto da Gordon e Lakoff e quindi da Searle, quando gli studiosi riuscirono a evidenziare l'esistenza di una precisa struttura, di uno schema, riguardante i procedimenti che un parlante poteva compiere per realizzare una richiesta indiretta; proponiamo alcuni esempi di tali procedimenti tratti dall'ATIS corpus.

1) Il parlante può esprimere una domanda riguardo la capacità dell'ascoltatore di eseguire una determinata attività:

A: Can you give me a list of the lights from Atlanta to Boston?

A: Could you tell me if Delta has a hub in Boston?

A: Would you be able to, uh, put me on flight with Delta?

2) Il parlante può far riferimento a un suo desiderio o a una gamma di bisogni riguardanti una determinata attività:

A: I want to fly from Boston to San Francisco

A: I would like to stop somewhere else in between

A: I'm looking for one way flights from Tampa to Saint Louis.

A: I need that for Tuesday.

A: I wonder if there are any flights from Boston to Dallas.

3) Il parlante può indurre l'interlocutore a svolgere un'azione prestabilita.

A: **Would you please repeat that information?**

A: Will you tell me the departure time and arrival time on this American flight?

4) Il parlante può chiedere se ha il permesso di ricevere i risultati connessi a una determinata azione.

A: May I get a lunch on flight U A two one instead of breakfast?

A: Could I have a listing of flights leaving Boston?

Basandosi su questi presupposti Searle ipotizzò che la sequenza di ragionamenti e trasformazioni effettuate dall'ascoltatore nell'elaborare la frase "*Can you give me a list of the flights from Atlanta to Boston?*" possono risultare le seguenti:

1) X mi ha posto una domanda riguardo una mia eventuale possibilità di suggerirgli una lista di voli.

2) assumo che X collabori nella conversazione (nel senso griceano del termine) e che la sua richiesta perciò abbia uno scopo.

3) X sa che io ho la capacità di suggerirgli tale lista, e che non c'è nessun ragionamento alternativo sul fatto che X abbia un interesse puramente teorico nella mia capacità di dargli questa lista.

4) l'enunciato di X ha quindi, probabilmente, una ulteriore funzione illocutoria. Quale può essere?

5) una condizione preparatoria affinché una direttiva venga portata termine implica che l'ascoltatore abbia la capacità di eseguire le azioni presupposte dalla direttiva.

6) X si è informato riguardo la mia capacità per l'azione sotto esame.

7) entrambi ci troviamo, inoltre, in una situazione conversazionale nella quale suggerire una lista di voli è un'attività comune e attesa.

8) nell'assenza di ogni altro plausibile atto illocutorio, X mi ha probabilmente richiesto, a tutti gli effetti, una lista di voli.

L'approccio inferenziale presenta una quantità di vantaggi. Prima di tutto, esprime perché una domanda come *Can you give me a list of flights from Boston?* è una ragionevole maniera di porre una richiesta indiretta mentre la frase *Boston is in New England* non lo è: la prima frase definisce una condizione già esistente per l'attività desiderata, e c'è una ragionevole catena inferenziale dalla condizione iniziale e preesistente, all'attività stessa.

L'approccio inferenziale è stato modellato da J. Allen, P. Cohen e C. R. Perrault e altri studiosi in una serie di importanti scritti riguardanti ciò che è stato chiamato il modello BDI (*belief, desire e intention*). Secondo tale modello, ispirato alle scienze dell'intelligenza artificiale, è possibile che un agente, cercando di trovare delle informazioni, utilizzi delle tecniche di pianificazione standard per chiedere a un parlante le informazioni desiderate.

L'approccio di questi studiosi fu applicato anche, naturalmente, e in modo più specifico, alla comprensione di effetti indiretti del linguaggio. Nel nostro studio inizieremo riassumendo le teorie formali di Perrault e Allen e la loro definizione di *ipotesi di desiderio* nel calcolo dei predicati.

Rappresentiamo l'enunciato "S crede alla proposizione P" come il predicato di second'ordine (con potenza due) $B(S, P)$ ¹¹⁵

Secondo tale annotazione l'enunciato *S knows that P* sarà rappresentato come KNOW(S,P) e definito anche come

$$\text{KNOW}(S,P) = P \wedge B(S,P)$$

¹¹⁵ Un predicato di potenza due, opera semplicemente su due variabili.

Analogamente alla definizione di *knows that* possiamo aggiungere quella di *knows whether* (KNOWIF)

In modo analogo ai due esempi presentati è possibile costruire un insieme di regole formali per definire lo svolgersi di un dialogo, portando in rilievo il ruolo delle intenzioni e delle supposizioni riguardanti una determinata frase o enunciato, e quindi riguardo ciò che si desidera venga svolto o compreso dall'eventuale interlocutore.

Un altro tipo di approccio è quello costruito su una interpretazione *cue-based* degli atti dialogici.

L'approccio precedentemente introdotto (l'approccio inferenziale) anche se piuttosto esaustivo nel gestire la gamma di casistiche è, ciò nonostante, piuttosto dispendioso per quanto riguarda il tempo necessario alla costruzione delle varie regole e annotazioni riferibili a una determinata frase. Tale approccio risulta in effetti orientato a produrre un modello del linguaggio AI-Complete: un problema AI-Complete come sappiamo è un problema che non può essere risolto veramente senza risolvere l'intero problema di realizzare una completa *Intelligenza Artificiale*.

Perciò per molte applicazioni, un metodo meno sofisticato ma più efficiente, e in grado di gestire dati specifici, può a volte essere sufficiente. Uno di questi metodi risulta una variante del metodo basato sugli idiomi, che abbiamo appena discusso.¹¹⁶

Ciò che caratterizza un modello *cue-based* è l'uso di una differente base di conoscenza per individuare un atto dialogico: possono essere utilizzate conoscenze a carattere lessicale, sintattico, prosodico, o estratte da strutture conversazionali. I modelli che tratteremo usano algoritmi supervisionati appartenenti alla categoria degli algoritmi per il *machine learning*, eseguiti su un corpus di dialoghi annotati manualmente. La gamma delle annotazioni utilizzate risulta come è ovvio dipendente dal sistema di sviluppo preso in considerazione.

Un esempio a proposito può risultare dal sistema di interpretazione degli atti di dialogo proposto da D. Jurafsky¹¹⁷ e che fa riferimento a tre sorgenti di informazione.

1) Parole e Sintagmi: *Please* o *would you*, nella lingua inglese potrebbero essere degli indici in grado di preannunciare una REQUEST (in italiano potremmo avere, *scusi, per favore, potrebbe, sarebbe così gentile da*)

2) Prosodia: accentare in particolar modo l'intonazione secondo le modalità di frasi interrogative è appunto un sistema per definire una QUESTION.

3) Strutture della conversazione: la parola di risposta "*hurrah*" a seguito di un proponimento è probabilmente un AGREEMENT, mentre nella frase propositiva "Un hurrah per il presidente" ha un'altra caratteristica.

Si consideri ad esempio le diverse realizzazioni della forma di richiesta annotata come CHECK. Essa in inglese può manifestarsi nelle cosiddette tag questions:

¹¹⁶ È necessario premettere che molti degli approcci, effettuati da svariati ricercatori possono avere delle caratteristiche diverse da quello di cui parleremo, sebbene con rilevanti analogie.

¹¹⁷ Jurafsky D. et al., "Automatic detection of discourse structure for speech recognition and understanding", in *Proceedings of the 1997 IEEE Workshop on Speech recognition and Understanding*, p. 88-95, Santa Barbara (1997).

A: and it's gonna take us also an hour to load boxcars right?
B: right

nelle frasi dichiarative:

A: and we have a powerful computer down at work.
B: Oh (laughter)
B: so, you don't need a personal one (laughter)?
A: No

in frammenti di domande:

A: Ehm, curve round slightly to your right.
B: To my right?
A: Yes.¹¹⁸

Possibili realizzazioni di questa funzione in LUCY sono evidenziate dalle seguenti trascrizioni automatiche.

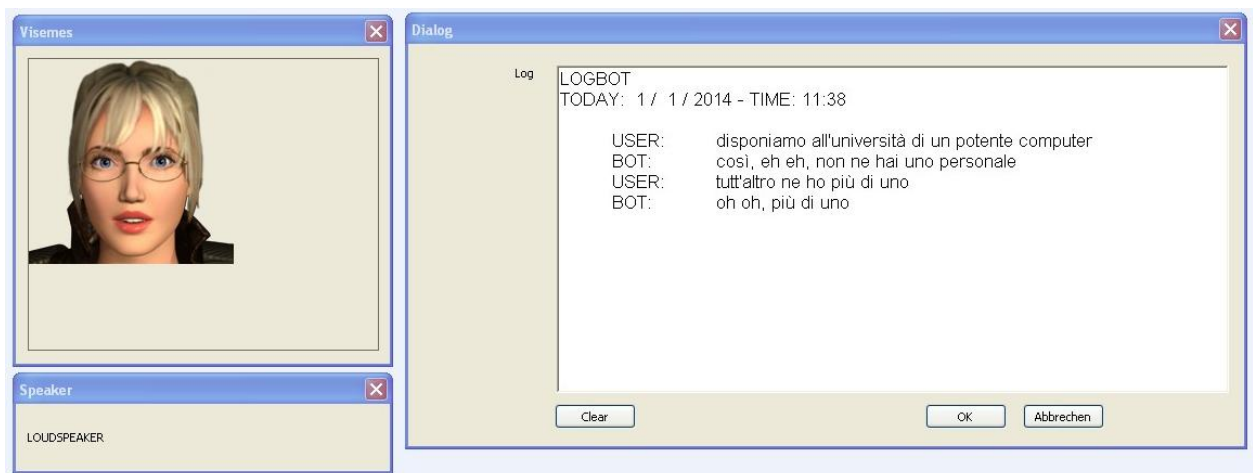


Fig. 55 - Realizzazione degli esempi precedenti. Utilizzo del “modulo RISATA”

¹¹⁸ Dal Map Task Corpus. L'HCRC Map Task Corpus è un insieme di 128 dialoghi registrati, trascritti, e annotati.

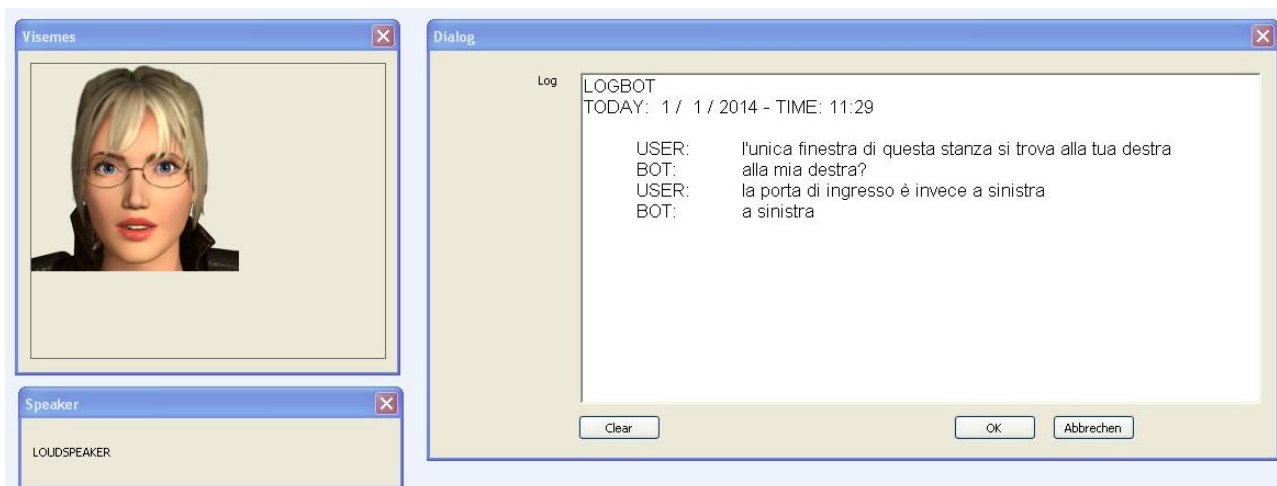


Fig. 56 - Esempio di funzione CHECK

L'uso di frasi dichiarative per aver conferma di quanto compreso dall'interfaccia dialogante contribuisce a creare una parvenza di partecipazione al dialogo secondo le modalità su cui abbiamo abbondantemente ragionato.

A proposito, è possibile naturalmente costruire un grafo dove le probabilità delle varie sequenze degli *speech acts* sono annotate e permettono di determinare quale atto si verifichi più probabilmente dopo un precedente atto, secondo le leggi della probabilità condizionata e il modello di distribuzione markoviano.

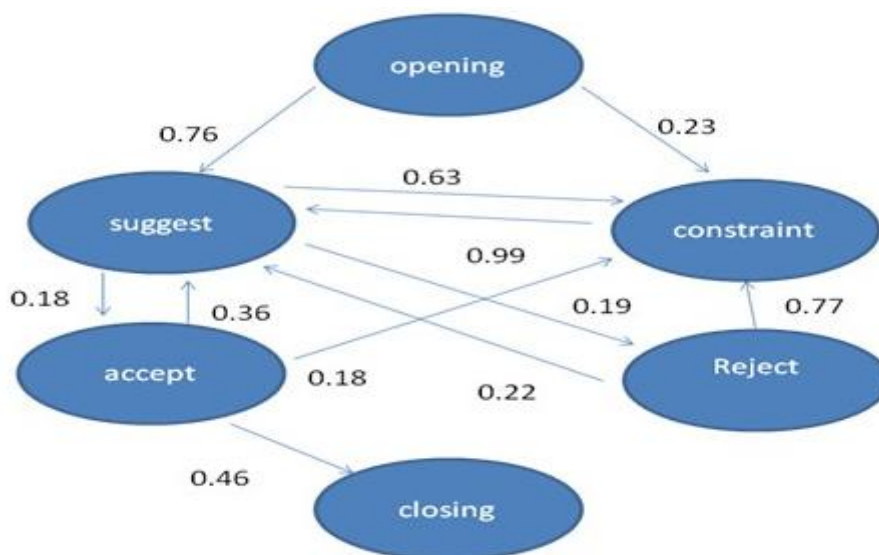


Fig. 57 - Un atto di dialogo secondo il modello markoviano (Woszczyna e Waibel, 1994)

In poche parole è possibile realizzare un modello per rappresentare lo sviluppo di una serie di atti di discorso secondo considerazioni eminentemente statistiche.¹¹⁹

Strutture dialogiche e coerenza

Precedentemente abbiamo descritto un approccio per determinare la coerenza di un dialogo basandoci su un insieme di relazioni di coerenza. Allo scopo di determinare se una relazione di coerenza risulti in grado di sostenere e *motivare* la frase, il sistema deve ragionare riguardo le restrizioni che la relazione fornisce all'informazione, nella frase espressa. Chiamiamo questo approccio: approccio "intenzionale".

Secondo tale tipo di approccio, una frase viene compresa come una sequenza di azioni, che richiedono una capacità dell'ascoltatore di dedurre informazioni a partire da ciò che viene espresso in superficie dal parlante, intuendo le intenzioni di quest'ultimo e evidenziandole secondo le presunte regole di coerenza. Al contrario dell'approccio informativo, l'approccio intenzionale viene applicato in maniera predominante proprio al dialogo. Alla base di questo approccio è l'idea che ogni discorso sia associato con uno scopo sottostante, e sottinteso dal parlante che ha iniziato il discorso (discourse purpose o DP). Parimenti, ogni segmento del discorso all'interno di un discorso ha uno scopo corrispondente, definito *Scopo del segmento di discorso* (DSP). Ogni DSP svolge un ruolo nell'ottenimento del DP del discorso. Le studiose Barbara Grosz e Candace Sidner,¹²⁰ che hanno proposto questo modello, distinguono tra Initiating Conversational Participant (ICP) e Other Conversational Participant (OCP) e cercano di codificare le molteplici intenzioni con cui un parlante può iniziare un discorso: dalla scelta della lingua con cui esprimersi, al desiderio di divertire, commuovere, o suscitare determinate impressioni.

Possiamo fornire esempi di alcuni scopi codificati:

- 1) il parlante vuole che l'ascoltatore esegua un qualche compito fisico.
- 2) il parlante desidera che l'interlocutore creda a un determinato fatto, o a una serie di fatti.
- 3) il parlante desidera che l'interlocutore creda che un fatto supporti un altro fatto.
- 4) il parlante desidera che l'interlocutore identifichi un oggetto (cioè un oggetto fisico, un oggetto immaginario, un evento, una sequenza di eventi).
- 5) il parlante desidera che l'interlocutore conosca alcune proprietà di un oggetto.

Grosz e Sidner propongono una diversa codifica dei rapporti di coerenza e cioè menzionano il rapporto di dominanza e di soddisfazione della "precedenza".

Anche in questo sistema ci troviamo comunque dinanzi al tentativo di classificare e modellare (spesse volte in strutture somiglianti) un dialogo secondo uno schema deterministico, gestibile da un eventuale algoritmo. Ma in sostanza come è possibile evincere dalla descrizione dei vari procedimenti di modellazione, come accadeva nel caso dei modelli di Schank e di Minsk di cui avevamo parlato a proposito dell'Information Extraction, la difficoltà di annotare manualmente ogni riferimento lascia supporre una utilità esclusivamente teorica di tali modelli e una loro applicabilità a contesti ristretti.

¹¹⁹ Nagata M., Morimoto T., "First steps towards statistical modeling of dialogue to predict the speech act type of the next utterance", *Speech Communication*, 15, pp. 193-203 (1994).

¹²⁰ Grosz B., Sidner C., "Attention, Intentions and the Structure of Discourse", *Computational Linguistics*, 12(3), pp. 175-204.

Dialogue Managers in agenti conversazionali

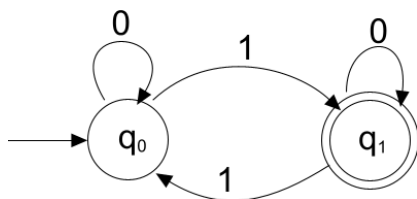
L'idea di una interfaccia dialogica e di un agente in grado di gestire determinate conversazioni risulta in ogni caso piuttosto accattivante e da questo presupposto, agenti come ELIZA, PARRY o SHRDLU sono tra i più conosciuti esempi di tecnologia operante nell'ambito del linguaggio naturale. Moderni esempi di agenti conversazionali includono i sistemi di informazione per la prenotazione di treni o aerei , guide turistiche per ristoranti, varie interfacce telefoniche, e attualmente anche realizzazioni nell'ambito delle automobili prodotte in serie.

Tra i componenti essenziali alla costruzione di questi agenti è necessario parlare del dialogue manager: il componente che cioè controlla il flusso del dialogo, e che decide come gli agenti dialoganti debbano procedere nell'ambito della conversazione, quali domande chiedere o asserzioni produrre, e quando intervenire nel flusso dialogico.

Esamineremo ora brevemente alcuni degli aspetti della progettazione di tali *dialogue manager*, discutendo e descrivendo alcuni semplici esempi basati su automi a stati finiti¹²¹ e regole di produzione, e anche alcuni altri sistemi più complessi basati su più sofisticati modelli e tecniche di pianificazione.

Naturalmente i più semplici dialogue manager sono, come già detto, basati su automi a stati finiti. Per esempio, si immagini un semplice sistema di interfaccia dialogica per la prenotazione di un viaggio in aereo: il compito iniziale dell'agente è quello di chiedere all'utente la città di partenza, la città di destinazione, l'orario e la preferenza dell'utente stesso riguardo una linea aerea o l'altra.

Un esempio di tale forma di interfaccia dialogica e delle sue funzioni principali all'interno di un ristretto contesto come quello presentato è definito nella figura qui di seguito riportata.



¹²¹ Un *automa a stati finiti* (ASF) o *macchina a stati finiti* o *FSA* (dall'inglese Finite State Automata) è un modello che permette di descrivere con precisione e in maniera formale il comportamento di molti sistemi. Nello specifico, con gli automi a stati finiti, si possono modellare tutti i sistemi che possiedono le caratteristiche di dinamicità (caratteristica di evolvere nel tempo passando da uno stato ad un altro) e discretezza (le variabili d'ingresso e gli stati del sistema da modellare possono essere espressi con valori discreti) . Una rappresentazione di un simile automa con due stati è la seguente:

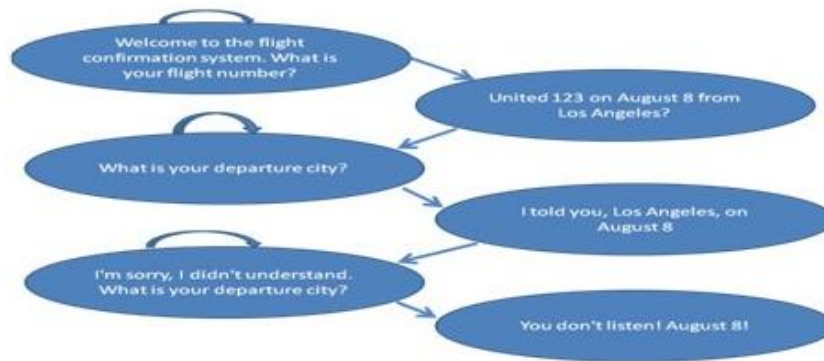


Fig. 58 - Automa a stati finiti di un Dialogue Manager

Gli stati dell'automa corrispondono alle domande che il *dialogue manager* chiede all'utente, e gli atti conseguenti corrispondono alle azioni da intraprendere, in relazione a ciò che l'utente risponde. Se ciò che l'utente dice non viene capito, l'automa ripropone la domanda, altrimenti prosegue nel suo flusso alla ricerca di informazioni che possono essere acquisite secondo un ordine predefinito. Sistemi che controllano completamente la conversazione in questa maniera sono chiamati sistemi a iniziativa singola. Mentre questo tipo di architettura è sufficiente per alcuni compiti in un contesto ristretto, ad esempio per realizzare un'interfaccia dialogica in un semplice gioco a quiz, è probabilmente troppo limitato per altri contesti più articolati, come anche nel caso stesso di sistemi per gestire le prenotazioni di un viaggio in aereo. Ovviamente ciò è dovuto al fatto che gli utenti utilizzino quasi sempre delle frasi più complesse di quelle individuate dagli esempi e che la gestione di tali domande risulta tutt'altro che deterministica come quella di un automa a stati finiti.

Uno dei primi importanti sistemi per svolgere tale compito risultò il GUS¹²² mentre uno dei più recenti è l'ATIS¹²³ insieme a una quantità di interfacce dialogiche utilizzate come guide turistiche generiche e operanti soprattutto nell'ambito della gastronomia.

Ritorniamo per il momento all'esempio di un sistema impostato per le prenotazioni di voli aerei: una delle mete principali di tale sistema è naturalmente quella di aiutare l'utente a individuare e prenotare il volo giusto in base alle sue esigenze. Tale sistema si potrebbe basare su una serie di frame o template da riempire con le informazioni specificate dall'utente, anche su richiesta del sistema stesso.

Il funzionamento del sistema Gus sembra proprio orientato in questo senso:

GUS: Hello. My name is GUS. I can help you to plan a simple trip by air. Where do you want to go?
 Client: I want to go to San Diego on May 28

¹²² Si veda GUS, A Frame-Driven Dialog System, Bobrow et al, 1977; GUS, sviluppato al centro di ricerca della Xerox di Palo Alto, è un sistema sviluppato per funzionare in un ambito molto ristretto: la conversazione di un agente di viaggio con un cliente che vuole prenotare un biglietto di andata e ritorno per una destinazione in California.

¹²³ L'ATIS, acronimo di *Automatic Terminal Information Service* è un sistema automatico di trasmissione delle informazioni generali, utili al pilota di un aereo, riguardanti le condizioni di volo. I messaggi possono essere trasmessi continuamente su frequenze dedicate. Contengono informazioni meteorologiche e operative destinate a essere utilizzate dai piloti che intendono utilizzare l'aeroporto da cui viene emesso l'ATIS. I messaggi tradizionali contengono l'identificazione dell'aeroporto, il vento dal suolo, la visibilità, e altre informazioni relative le condizioni e i parametri di volo. I messaggi ATIS vengono aggiornati ogni 30 minuti o quando vi sono cambiamenti significativi dei parametri.

GUS: What time do you want to leave?
 Client: I must be in San Diego before am
 ...
 GUS: Would you like the flight that leaves at 7.45 am
 Client: That's fine.
 GUS: I have confirmed the following flight: P.S.S. flight 307 on Friday May 30 from San Diego to San Jose that leaves at 7.45 pm and arrives at 9.30 pm. Thank you for calling. Goodbye.

Tab. 12 - Trascrizione di un dialogo con GUS

Alcuni dei template utilizzabili per ottenere le informazioni possono essere costruiti su specifiche domande da chiedere all'utente; come nel caso della Information Retrieval chiameremo *slot* le informazioni da ottenere attraverso i template utilizzati.

SLOT	OPTIONAL QUESTION
From_Airport	From what city are you leaving?
To_Airport	Where are you going?
Dep_time	When would you like to leave?
Arr_time	When do you want to arrive?
Fare_class	
Airline	
Oneway	

Tab. 13 - Esempio di Template

Un tale semplice *dialogue manager* può soltanto domandare all'utente di riempire i template con una serie di risposte fino a quando non ha le informazioni sufficienti per eseguire una ricerca all'interno del database e riportare poi il risultato all'utente.

È necessario premettere che in genere è preferibile non associare in maniera biunivoca una domanda a uno slot da riempire poiché è spesso volte opportuno non subissare l'utente di domande troppo specifiche e a volte superflue.

Ciò nonostante il sistema deve essere in grado di riempire gli eventuali slot qualora l'utente lo richieda espressamente.

In ogni caso, anche tale breve esempio richiederebbe più della semplice struttura basata su un *singolo template* alla quale abbiamo appena accennato. Si potrebbe presentare cioè, più di un solo volo corrispondente alle richieste specifiche dell'utente. In questi casi, all'utente sarà presentata una lista di scelte, o attraverso lo schermo video, oppure attraverso una interfaccia vocale, ascoltando le proposte oralmente. Un sistema basato sui template può allora disporre di un altro tipo di template (un sottoinsieme collegato al template principale) che ha slot per identificare elementi appartenenti alla lista dei voli, o ad altre ulteriori liste (“quanto costa il primo volo?” O “il secondo volo è un volo diretto?”). Altri template possono avere delle informazioni generali riguardo l'itinerario (con domande tipo “quale compagnia compie un itinerario da Milano a Roma?”), informazioni riguardanti le pratiche di acquisto dei biglietti (per domande come “quanti giorni prima devo prenotare il volo allo scopo di avere una tariffa agevolata?”) Oppure ancora altri tipi di richieste come prenotazioni di auto o di hotel.



Fig. 59 - Esempio

Poiché gli utenti possono consultare l'interfaccia secondo molteplici *template* di riferimento, e poiché essi possono rispondere in modo indipendente alle richieste espresse dal sistema, il sistema deve essere capace di “disambiguare” quale slot di quale template, può essere riempito con un determinato input e di conseguenza condurre il dialogo facendo riferimento al modello di quel template.

Nel mio sistema sfrutto proprio tale approccio per convogliare il flusso dell'informazione e consultare i pattern più probabili per il tipo di discorso in questione. In base all'argomento definibile dal tipo di domande si procede a scegliere i template all'interno di una directory corrispondente a tale argomento fino a che, quest'ultimo non viene cambiato o si desidera un approfondimento. Nel caso dell'esempio viene utilizzata una subdirectory dell'argomento in questione e così via.

I seguenti codici di markup, ed altri analoghi, gestiscono sostanzialmente il flusso del discorso nella mia architettura:

```
<category><pattern><filepattern>parlare</filepattern>
  <c> parliamo di */c</c></pattern>
  <template><action>argomento</action><filetemplate>vediamo</filetemplate>
</template>
</category>
```

L'action “argomento” usa il parametro che sostituisce la `<star>`¹²⁴ e utilizza i template della directory indicizzata da quest'ultimo.

Attraverso tale procedimento ho potuto ridurre il numero di template consultati dopo ogni domanda e di conseguenza anche i tempi di una eventuale risposta.¹²⁵

¹²⁴ Come vedremo nelle spiegazioni in dettaglio la “variabile” `<star/>` contiene l'argomento della frase pronunciata dall'utente. Ad esempio nella frase “parliamo di *fisica*”, il token “fisica” andrà a riempire lo slot corrispondente al template `star`; nella fattispecie `<star>fisica</star>`

¹²⁵ In dettaglio l'attraversamento dell'albero di ricerca è effettuato con un criterio di *breadth search* (tecnica di attraversamento di una struttura ad albero), dal momento che allo stato attuale, e per lo scopo esemplificativo propostomi, non ho previsto grande profondità nella ramificazione dell'albero (o dell'eventuale grafo).

Possiamo quindi definire questo tipo di approccio template-based, come un sistema a regole di produzione. Differenti tipi di input ci indirizzano verso diversi insiemi di risposte costruite su diversi pattern.

Ovviamente tale approccio, come abbiamo più volte ripetuto, è estremamente efficace nell'ambito di settori limitati (il sistema GUS funzionava soltanto per le prenotazioni aeree verso la California): riguardo una situazione o un argomento con queste caratteristiche è possibile "prevedere" le domande più probabili e di conseguenza gestire le risposte e il loro flusso in maniera, per certi versi, deterministica, attraverso una rete di suggerimenti e ipotesi.

Malgrado ciò, per l'eventuale attuazione, è necessario un tempo piuttosto lungo dovuto alla caratteristica delle varie annotazioni, spesse volte non adattabili ai vari template.

Si consideri ora ad ulteriore esempio il seguente scambio dialogico:

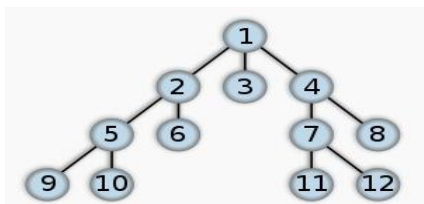
- A1: And what time would you like to leave Pittsburgh?
- B1: Uh hmm I don't think there's many options for non-stop.
- A2: Right. There's three non-stops today.
- B2: What are they?
- A3: The first one departs PGH at 10:00am...

Nella frase presente in B1 il cliente pone una sorta di domanda indiretta, individuata dal sistema, che a sua volta effettua delle proposte focalizzando l'attenzione sulla priorità della ricerca di voli senza scalo. La gestione delle domande indirette è più complessa ma può spesso essere effettuata con gli stessi criteri della gestione di domande dirette. È necessario cioè individuare i *pattern* che producono queste domande ed estrarre le richieste come in una domanda normale: in questo caso "I don't think _" ma poteva anche essere "maybe _", "I was thinking that _" e altre possibilità analoghe.

Oltre a queste complicazioni, in tali sistemi risulta estremamente importante gestire la ripetizione delle informazioni senza appesantire il discorso, ma garantendo eventuali controlli di validità in grado di limitare errori dovuti a una mancata comprensione, ad esempio del riconoscimento vocale, oppure legata ad altri fattori.

Allo scopo di affrontare tali problematiche, molti sistemi utilizzano più sofisticati *dialogue manager* basati sull'architettura BDI (belief, desire, intention) che abbiamo già descritto. Questi sistemi sono essenzialmente strutturati su un modello di regole logiche e gestisce una conversazione come una sequenza di azioni da pianificare.

Un esempio di tali sistemi è il TRAINS un'interfaccia dialogante nella quale il compito dell'agente è quello di collaborare alla gestione di un sistema di trasporti in un ipotetico



Ordine dei nodi nell'attraversamento breadth-search di un grafo.

microsistema: l'utente e il sistema possono collaborare nella pianificazione dello spostamento di un container da una città a un'altra. L'utente osserva una mappa geografica con l'indicazione delle varie città di partenza e destinazione, e le relative connessioni ferroviarie. Deve gestire una serie di problemi logistici, prendendo in considerazione diversi parametri ambientali riguardanti il trasporto e ovviamente le connessioni esistenti

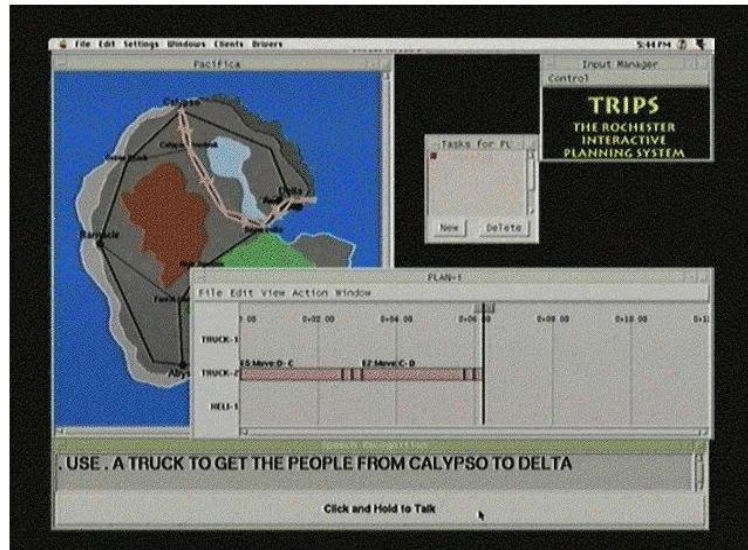


Fig. 60 - Schermata del sistema TRAINS

Il dialogue manager del TRAINS mantiene la gestione del flusso della conversazione e provvede a individuare gli scopi di tale conversazione. Per i suoi intenti il sistema deve modellare i diversi stati del dialogo, le sue intenzioni, le richieste dell'utente, i presupposti e i reali obiettivi delle sue asserzioni. Il sistema effettua difatti una analisi semantica di tali asserzioni e dispone di un generatore di frasi per produrre le "risposte" destinate all'utente.

User[1]: I need to get to Detroit, Syracuse and Montreal.
Syst[2]: OK. [highlights the three cities]
User[3]: Let's send the trains from Chicago to Detroit.
Syst[4]: Ok. [draws route from Chicago to Detroit via Toledo] But the terminal at Toledo is delaying traffic due to bad weather. Trains will take an additional five hours to move from there. [highlights Toledo red]
User[5]: That's ok. Now let's go from Atlanta up to Syracuse.
Syst[6]: Please tell me a route to use to get from Atlanta to Syracuse [highlights engine at Atlanta and city Syracuse]
User[7]: Go via Charleston and Pittsburgh.
Syst[8]: Ok. [draws route via Cincinnati] But traffic through Cincinnati is being delayed due to construction.
User[9]: Ok, let's go via Charleston and Buffalo instead.
Syst[10]: No problem. [draws route]
User[11]: That's great.
Syst[12]: I hope it was helpful.
User[13]: Finally, I want to go from Montreal to Philadelphia.
Syst[14]: Ok. [draws route]
User[15]:

Tab. 14 - Trascrizione di un dialogo con TRAINS

La struttura del sistema TRAINS è definita nel seguente diagramma, che può essere esemplificativo di qualsiasi altra architettura di una interfaccia dialogante:

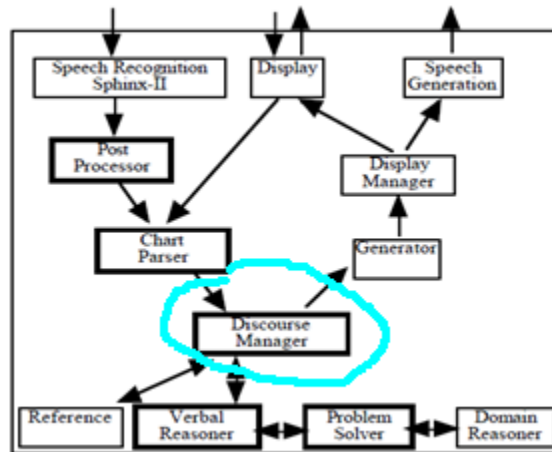


Fig. 61 - Architettura del sistema TRAINS

In riferimento al Discourse Manager un possibile algoritmo del modulo è il seguente:

DIALOGUE MANAGER

While la conversazione non è finita
If l'utente ha completato un turno di richiesta e espresso una frase (REQUEST)
Then il sistema interpreta la frase
If il sistema deve soddisfare una richiesta (OBLIGATION)
Then il sistema soddisfa la richiesta
Else if il sistema deve realizzare un atto conversazionale
Then impiega il generatore di testo per la frase da produrre nella conversazione
Else if alcuni elementi della conversazione non sono inerenti
Then cerca chiarimenti
Else if le richieste sono inconsistenti
Then cerca chiarimenti
Else if scopi della conversazione iniziale non sono raggiunti ma raggiungibili
Then il sistema continua a perseguire tali scopi
Else finisce il proprio turno o cerca di terminare la conversazione (REJECT)
Else if non è il turno di nessuno
Then prende il turno
Else if dopo una lunga pausa
Then prende il turno

Tab. 15 - Possibile algoritmo di un dialogue manager.

Esaminiamo alcune delle caratteristiche principali del sistema TRAINS in maggiore dettaglio: dinanzi a ogni richiesta dell'utente (REQUEST) il sistema deve definire una OBLIGATION volta a soddisfare tale richiesta o a respingerla qualora inattuabile (REJECT).

Quindi deve riflettere sui propri obiettivi e le modalità per raggiungerli e creare una appropriata pianificazione.

L'idea di atti conversazionali impiegati da tale sistema può essere compresa attraverso il seguente esempio. Supponiamo l'utente ponga la richiesta:

A1: I want to go to Pittsburgh in May

A questo punto della conversazione, il turno di parlare è assegnato all'agente che come meta ha quella di individuare un itinerario soddisfacente per raggiungere Pittsburg nel mese di maggio dell'anno corrente (l'informazione è aggiunta in modo implicito).

La tabella (16) definisce alcuni dei parametri di stato del sistema, prima della frase in questione: la lista degli obblighi, la lista degli *speech acts* da trasmettere al *text generator*, la lista degli obiettivi del discorso e se il sistema deve intervenire o no nel dialogo, ossia se è il suo turno.

Discourse obligations:	NONE
Turn holder:	system
Intended speech acts:	NONE
Unacknowledged speech acts:	INFORM-1
Discourse goals:	get-travel-goal, create-travel-plan

Tab. 16 - Stato del sistema in seguito a una richiesta

Dopo la frase di partenza il dialogue manager decide di inserire due atti conversazionali al dialogo: il primo per definire il riconoscimento dell'atto informativo INFORM pronunciato dall'utente e il successivo per chiedere delle specifiche al sistema riguardanti l'itinerario da seguire. Il terzo parametro è perciò mutato nel seguente modo:

Intended speech acts: REQUEST-INFORM-1, ACKNOWLEDGE-1,

L'assegnazione è effettuata partendo dal presupposto che l'algoritmo insito nel sistema induce alla formulazione di una domanda allo scopo di raggiungere gli scopi informativi prefissati nel parametro *Discourse goals*.

Attraverso un *text generator* sarà infine possibile produrre la seguente singola frase comprendente entrambe le istanze informative:

A1: And, what day in May do you want to travel?

Nella ripetizione del termine "may", il mese oggetto di interesse, si può notare il tentativo di rafforzare il *grounding* del dialogo.

La richiesta dell'informazione è ottenuta attraverso la domanda "che cosa".

Passiamo adesso ad esaminare la frase pronunciata dal cliente B1. Ricordiamo che tale frase è una domanda indiretta con cui si chiede all'agente di controllare eventuali voli senza scalo.

A1: And what time would you like to leave Pittsburgh?

B1: Uh hmm I don't think there's many options for non-stop.

Supponiamo ora che l'interprete degli atti di dialogo individui correttamente in B1 una richiesta di informazioni. Lo stato del sistema risulterà ulteriormente cambiato secondo il seguente schema.

Discourse obligations:	address (REQUEST-INFORM-3)
Turn holder:	system
Intended speech acts:	NONE
Unacknowledge speech acts:	REQUEST-INFORM-3
Discourse goals:	get-travel-goal, create-travel-plan

Il sistema dovrà innanzitutto rispondere alla richiesta dell'utente richiamando la procedura di pianificazione per scoprire se ci sono voli senza scalo. Quindi deve rispondere alla domanda indiretta, ma deve anche dare credito all'osservazione dell'utente. Nei casi di richiesta diretta, la risposta denota una sufficiente intesa che funge come una sorta di *grounding*. Nei casi invece di richiesta diretta, con riferimento esplicito a quanto espresso dall'utente, il *grounding* può essere opzionale; una possibile soluzione può essere data dalla seguente risposta che comprende come è possibile denotare due atti di dialogo.

A4: Right. There's three non-stops today.

Nel primo si sottolinea la correttezza dell'osservazione del parlante-utente, nel secondo si produce una risposta in grado di soddisfare quest'ultimo.

Altre opzioni possibili in questo genere di sistema risultano incentrate sulla capacità di generare e gestire dei sotto dialoghi, dal loro inizio fino ai vari sviluppi di questi ultimi.

Difatti il sistema gestisce le coppie di frasi adiacenti in maniera piuttosto complessa ed esemplare. Dopo la frase (User3) della trascrizione riportata inizialmente al suggerimento del parlante, il sistema dapprima valuta quest'ultimo, e quindi esprime delle problematiche, iniziando una specie di sotto dialogo.

User[3]:	Let's send the trains from Chicago to Detroit.
Syst[4]:	Ok. [draws route from Chicago to Detroit via Toledo] But the terminal at Toledo is delaying traffic due to bad weather. Trains will take an additional five hours to move from there. [highlights Toledo red]

Altra prerogativa del sistema è la valutazione del contesto e una chiara definizione dello sviluppo del discorso. Sempre in riferimento all'esempio iniziale l'utente propone (nella frase User9) una alternativa e il sistema, valutando appunto il contesto, capisce che il riferimento è alla via attraverso Buffalo e cancella l'opzione Cincinnati precedentemente proposta.

Syst[8]:	Ok. [draws route via Cincinnati] But traffic through Cincinnati is being delayed due to construction.
User[9]:	Ok, let's go via Charleston and Buffalo instead.
Syst[10]:	No problem. [draws route]
User[11]:	That's great.

9. Esempi di Chatbot

*Eliza*¹²⁶

ELIZA è un chatterbot sviluppato nel 1966 da Joseph Weizenbaum al MIT di Boston.

Fu chiamato così da Eliza Doolittle, la fioraia dall'eloquio incolto e dialettale, protagonista della commedia *Pigmalione*¹²⁷ di George Bernard Shaw che, edotta per una sorta di scommessa dal professor Higgins, impara il raffinato modo di esprimersi delle classi più agiate.

Il programma di Weizenbaum imita quasi in forme parodiche un terapeuta rogersiano,¹²⁸ in buona parte rispondendo all'ipotetico paziente che interagisce con il paziente mediante domande ottenute dalla riformulazione delle affermazioni del paziente stesso. Così, per esempio, alla frase "Mi fa male la testa" il programma può ribattere con "Perché dici che ti fa male la testa?" oppure la risposta a "Mia sorella mi odia" potrebbe essere "Chi altro nella tua famiglia ti odia?"

ELIZA può essere considerato uno dei più popolari programmi di intelligenza artificiale di tutti i tempi, e alcune sue versioni esistono sia sotto forma di programmi per il WEB o su diversi sistemi operativi.

Weizenbaum disse che Eliza è una parodia delle "domande di uno psicoterapista all'inizio di un intervento psichiatrico". Lo studioso scelse la psicoterapia "per evitare il problema di dare al programma una vera conoscenza": la seduta terapeutica è una di quelle poche situazioni in cui un essere umano può rispondere ad una affermazione con una domanda che parte da quella poca conoscenza del soggetto in discussione. Per esempio, in tale contesto, alla domanda "Chi è il tuo compositore preferito?" può essere accettabile che si risponda con la domanda "Che ne dici di parlarci del tuo compositore preferito?" o "Questa domanda ti interessa?"

Eliza non è costruito secondo una ben specifica teoria di comprensione e di produzione del linguaggio, ma serve come eccellente esempio all'interno del nostro discorso, poiché il suo comportamento, inizialmente, è davvero ragguardevole. Il sistema recita proprio il ruolo di uno psicoterapeuta e, per ottenere i migliori risultati, l'utente dovrebbe svolgere in corrispondenza il ruolo di un paziente.

Abbiamo presentato la trascrizione di un dialogo con questo sistema.

USER	ELIZA
Men are all alike.	In what way?
They're always bugging us about something or other.	Can you think of a specific example?
Well, my boyfriend made me come here.	Your boyfriend made you come here.

¹²⁶ Ci sono diversi programmi basati su ELIZA in diversi linguaggi. Inizialmente ELIZA fu incluso in alcuni dei primi giochi per computer, con diversi tipi di interfaccia grafica. Una versione eseguibile scritta in JAVA è presente all'indirizzo <http://www.chayden.net/eliza/Eliza.html>. Un'altra versione disponibile sul Web in Javascript è all'indirizzo <http://www.masswerk.at/elizabot/>.

¹²⁷ G. B. Shaw, *Pigmalione*. Commedia in cinque atti andata in scena per la prima volta a Vienna nel 1913.

¹²⁸ Carl Ramson Rogers (1902 - 1987) è stato uno psicologo statunitense, fondatore della terapia non direttiva e noto in tutto il mondo per i suoi studi sul *counseling* e la psicoterapia all'interno della corrente umanistica della psicologia.

USER	ELIZA
He says I'm depressed much of the time.	I am sorry to hear you are depressed.
It's true, I am unhappy.	Do you think coming here will help you not to be unhappy?
I need some help; that much seems certain.	What would it mean to you if you got some help?
Perhaps I could learn to get along with my mother.	Tell me more about your family.
My mother takes care of me.	Who else in your family takes care of you?
My father.	Your father.
You are like my father in some ways.	What resemblance do you see?

Tab. 17 - Dialogo con ELIZA

La scelta di non tradurre le varie frasi del sistema dialogante con il suo interlocutore è in questo caso necessaria poiché il funzionamento di Eliza è incentrato proprio, come vedremo, nelle caratteristiche della lingua inglese. Difatti il chatbot è sviluppato a partire da un particolare database di parole chiave. Per ogni parola chiave, il sistema memorizza un valore intero, e una sequenza di stringhe che vengono confrontate con l'input preposto, e che definiscono una corrispondenza con uno specifico output. L'algoritmo è il seguente: data una frase *S* è necessario trovare una parola chiave al suo interno il cui pattern corrisponde con *S*. Se esistono molteplici parole chiave, si sceglie quella con il valore intero più alto. Si utilizzano poi le specifiche che definiscono l'output associato con le parole chiave per generare la frase richiesta dalla specifica domanda. Se non ci sono parole chiave è possibile generare delle frasi che servano per proseguire la conversazione, come ad esempio "prosegui", "puoi dirmi di più".

Presentiamo qui di seguito un frammento di alcune parole chiave. In questo database un pattern consiste di parole e variabili. Il prefisso "?" dinanzi a una lettera indica una variabile, che può corrispondere a qualsiasi sequenza di parole. Per esempio il pattern:

?X are you ?Y

Può corrispondere alla frase "Why are you looking at me?", dove la variabile ?X corrisponde a "Why" e "?Y" corrisponde a "looking at me".

Word	Rank	Pattern	Outputs
Alike	10	?X	In what way? What resemblance do you see?
Are	3	?X are you ?Y	Would you prefer it if I weren't ?Y?

Word	Rank	Pattern	Outputs
	3	?X are ?Y	What if they were not ?Y?
Always	5	?X	Can you think of a specific example? When? Really, always?
What	2	?X	Why do you ask? Does that interest you?

Tab. 18 - Corrispondenze di parole chiave

Le specifiche dell'output possono utilizzare le stesse variabili. In questo caso, Eliza inserisce le parole che corrispondono alle variabili nell'input, all'interno dell'output dopo aver attuato piccoli cambiamenti, nella scelta dei pronomi (per esempio, sostituendo "me" con "you").

Con i pattern a disposizione, se le specifiche dell'output sono:

Would you prefer it if I weren't ?Y

La regola genererà la seguente risposta: *"Would you prefer it, if I weren't looking at you?"*.

Quando il database elenca le molteplici specifiche per l'output di un determinato pattern, Eliza seleziona una differente specifica ogni volta che quella particolare regola associata alla parola chiave è usata, riuscendo perciò a evitare ripetizioni all'interno della conversazione. Usando queste regole, in riferimento alla trascrizione presentata, si può comprendere come Eliza abbia prodotto le prime battute di dialogo.

Men are all alike.	In what way?
--------------------	--------------

Tab. 19.

Eliza genera la prima risposta dal primo output della parola chiave "alike" e la seconda risposta dal primo output della parola chiave "are".

La descrizione che abbiamo dato definisce in sostanza le parti essenziali del programma considerato. Possiamo perciò convenire che il programma non capisce la conversazione a cui sta partecipando. Piuttosto, il chatbot risulta un insieme di stratagemmi operanti all'interno di un sistema. Ciò nonostante, è indubbio che Eliza almeno all'interno di contesti specifici funzioni piuttosto bene. Molteplici sono le ragioni che determinano queste notevoli "prestazioni". Forse la ragione principale consiste nel fatto che quando le persone ascoltano o leggono una sequenza di parole individuabile come una frase, esse attribuiscono un *significato* alla frase e presumono che la persona (o la macchina) che ha prodotto quella frase abbia inteso quel significato in maniera specifica. Le persone sono estremamente abili

nel distinguere i significati delle parole e nell'interpretare le frasi all'interno di un contesto. Eliza sembra comportarsi in maniera intelligente poiché noi stessi utilizziamo la nostra capacità di comprensione per dare un senso a ciò che il chatbot dice.

Infatti, secondo quanto scrive Weizenbaum nel suo libro *Computer Power and Human reason*:

ELIZA [il programma dal quale fu ricavato Il dottore] creava nella mente delle molte persone che ci conversavano l'illusione persistente di capire realmente... Spesso esse chiedevano il permesso di conversare col sistema in privato e dopo un po' sostenevano, nonostante le mie spiegazioni, che veramente la macchina li aveva capiti.¹²⁹

Ci sono poi altre importanti caratteristiche e impostazioni del programma che contribuiscono a sostenere l'illusione di un comportamento intelligente. Per esempio, il sistema non ha bisogno di una particolare conoscenza del mondo poiché non deve mai esporre un reclamo, supportare una argomentazione, o rispondere a una ben specifica domanda.

Piuttosto, finisce per chiedere sempre una serie di domande che fanno da contraltare alle domande presentate in prima istanza. Naturalmente ciò sarebbe inaccettabile al di fuori di un contesto come quello tra un paziente e il suo psicoterapeuta. Eliza evade tutte le domande dirette rispondendo con un'altra domanda, come ad esempio "*Why do you ask?*". È praticamente impossibile forzare il programma a dire qualcosa di concreto riguardo un determinato e specifico argomento.

Anche in tale situazione ristretta, comunque, è relativamente facile dimostrare che il programma non capisce il linguaggio umano. A volte produce delle risposte completamente senza senso. Per esempio, alla domanda "*Necessity is the mother of invention*", potrebbe rispondere con un "*Tell me more about your family*", basandosi nei suoi pattern per la parola "mother". Inoltre, poiché Eliza non ha alcuna conoscenza riguardante la struttura del linguaggio, accetta frasi non grammaticali alla stessa maniera di frasi realmente accettabili dalla grammatica e dotate di senso. Digitando "*Green the adzabak wsfffrat are the a ran four*" Eliza risponderà con qualcosa tipo "*What if they were not the a ran four?*"

Inoltre, a proposito del progredire della conversazione, il programma non conserva memoria del contenuto della stessa. Inizia nel proporre domande che sembrano inappropriate alla luce di primi scambi di conversazione, e le sue risposte in genere iniziano mostrando una certa incapacità di focalizzare un determinato argomento.

Ora supponiamo di aver bisogno di costruire un programma per l'elaborazione del linguaggio naturale a breve termine. Se iniziano costruendo un modello generale di come avviene la compressione del linguaggio, il nostro programma non sarà completo probabilmente, nell'arco di tempo prestabilito, e così le sue prestazioni risulteranno deludenti nel breve termine. Un sistema come Eliza, ciò nonostante, può facilmente produrre un comportamento come quello che abbiamo appena discusso con relativamente poche linee di codice, comportamento che apparirà di gran lunga molto superiore all'altro sistema coinvolto.

È proprio con tale consapevolezza che ho intrapreso il lavoro presentato in questa ricerca: da una parte ho cercato di raggiungere risultati apprezzabili nel breve termine, di costruire

¹²⁹ Weizenbaum Joseph, *Computer Power and Human Reason: From Judgement to Calculation*, p. 189, W.H. Freeman and company, San Francisco (1976).

cioè un sistema, concretamente utilizzabile anche all'interno di un contesto di interessi limitato.

Ciò nonostante, non ho mai perso di vista la necessità di realizzare e costruire una ben definita teoria del linguaggio sulla quale basare gli sviluppi successivi della realizzazione di una interfaccia. Ho in un certo senso proceduto su due fronti, cercando di congiungere i risultati ottenuti in entrambi.

A proposito, come evidenzierò meglio nelle conclusioni, dalla mia esperienza di ricerca ho potuto constatare che all'interno di un contesto ben specifico è possibile produrre un sistema in grado di comportarsi intelligentemente e superare le carenze presentate da un programma come Eliza.

Mi riferisco a un esempio concreto: supponiamo di voler costruire un sistema dialogante in grado di rispondere a domande incentrate su un determinato romanzo. Alla luce del contesto specifico relativo al romanzo preso in considerazione è possibile realizzare una struttura in grado di rispondere con esattezza a domande come “chi è il protagonista”, o anche “dove è ambientato il romanzo”. Potremmo cioè realizzare un sistema supervisionato che già prevede le possibili domande relative al contesto specifico. Il sistema potrebbe rispondere con argomentazioni ben strutturate, senza evadere qualsivoglia domanda, e anzi suggerendone altre, proprio nell'ambito della supervisione.

Il problema, a mio avviso sorge quando si sposta il raggio del nostro interesse al di fuori del contesto preso in analisi; ad esempio, considerando un diverso romanzo, sarebbe necessario ricostruire la struttura precedentemente realizzata, proprio con il nuovo romanzo, iterando la supervisione, anche per le stesse domande, con notevole dispendio di tempo, e attività lavorativa. Sarebbe possibile evitare tale nuova supervisione, qualora il sistema fosse in grado da sé, di recuperare le informazioni sulla base di uno specifico schema, o modello. Per questo motivo ritengo che le discipline legate alla cosiddetta *Information Extraction* coadiuvate da tecniche appropriate di *Machine Learning* siano alla base della costruzione di un qualsiasi sistema dialogante. Per costruire delle risposte, cioè, è necessario individuare dove tale risposte sono presenti, e attuare delle strategie che permetta al sistema non supervisionato di funzionare autonomamente.

PARRY

Parry venne sviluppato nel 1972 dallo psichiatra Kenneth Mark Colby dell'Università di Stanford.

Simulava il dialogo con un individuo paranoico le cui risposte bizzarre erano spesso basate su astruse concettualizzazioni e fraintendimenti.

Negli anni settanta venne eseguito un esperimento con una sorta di variazione del test di Turing. Un gruppo di psichiatri analizzarono le risposte di alcuni computer che eseguivano PARRY e di alcuni pazienti che erano collegati a questi ultimi tramite una telescrivente. A un diverso gruppo di 33 psichiatri vennero mostrate le trascrizioni delle conversazioni avvenute. Ai due gruppi venne poi chiesto di identificare quali dei “pazienti” erano umani e quali invece dei programmi. Gli psichiatri furono in grado di dare una corretta interpretazione solo nel 48 per cento dei casi, una percentuale vicina alla probabilità di una scelta casuale.

PARRY ed ELIZA vennero confrontati diverse volte. Il più famoso di questi incontri avvenne a distanza, su ARPANET la prima “versione” di internet negli Stati Uniti. La trascrizione di uno di tali dialoghi è la seguente:

ELIZA: Good Evening. Tell me your problems. Please terminate input with a period or a question mark.

PARRY: People get on my nerves sometimes.

ELIZA: I am not sure I understand you fully.

PARRY: You should pay more attention.

ELIZA: Suppose you should pay more attention.

PARRY: You're entitled to your own opinion.

ELIZA: What makes you think I am entitled to my own opinion?

...

PARRY: I went to the races.

ELIZA: Please go on.

PARRY: Do you know anything about bookies?

ELIZA: What makes you think I know anything about bookies?

PARRY: A bookie didn't pay me off once.

ELIZA: What does that suggest to you?

PARRY: I beat up the bookie who did not pay off, and I thought he might use his friends in the underworld to get even with me

...

PARRY: In my opinion you have a one-track mind.

ELIZA: What makes you think I have a one-track mind?

PARRY: I used to go to the track every week.

ELIZA: What does that suggest to you?

PARRY: I avoid racetracks now because the races are fixed.

ELIZA: Is that the real reason?

PARRY: I'd rather not discuss it.

ELIZA: What else would you like to discuss?

PARRY: That's something I don't want to go into.

ELIZA: Does that trouble you?

PARRY: I went to the races.

...

PARRY: The way you repeat yourself is getting ridiculous.

ELIZA: Suppose you repeat myself is getting ridiculous.

...

PARRY: You are a real nag.

ELIZA: Does it please you to believe I am a real nag?

...

PARRY: Goodbye.

ELIZA: It's been my pleasure, that's \$399.29 please.

Ben diverso è invece il tono i conversazione con un esser umano

INTERVIEWER: How are you today?

PARRY: OK.

INTERVIEWER: What is your name?

PARRY: My name is Frank Smith.

INTERVIEWER: How old are you?

PARRY: Twenty-eight.

INTERVIEWER: Are you married?

PARRY: No.

INTERVIEWER: What sort of work do you do?

PARRY: I work in the Post Office.

INTERVIEWER: Why are you in the hospital?

PARRY: I shouldn't be here.

INTERVIEWER: Who brought you here?

PARRY: The police.

INTERVIEWER: What trouble did you have with the police?
 PARRY: Cops don't do their jobs.
 INTERVIEWER: What do they do instead?
 PARRY: Cops arrest the wrong people.
 INTERVIEWER: They arrest the right ones sometimes.
 PARRY: The law should get those Italian crooks.
 INTERVIEWER: Are you Italian?
 PARRY: My background is British, but I was born in this country.
 INTERVIEWER: Do you know any Italian crooks?
 PARRY: I try to avoid the underworld.
 INTERVIEWER: Are you afraid of the underworld?
 PARRY: The mafia are the worst gangsters.
 INTERVIEWER: Are you connected with the mafia?
 PARRY: You know, they know me.
 INTERVIEWER: How do they know you?
 PARRY: They spy on me.
 INTERVIEWER: Why do they?
 PARRY: They want to get false evidence to frame me.
 INTERVIEWER: What do they have against you?
 PARRY: They are out to get me.
 INTERVIEWER: What for?
 PARRY: I have incriminating evidence against the mafia.
 INTERVIEWER: Are you sure of all this?
 PARRY: You don't believe me?[8]¹³⁰

SHRDLU

Il programma SHRDLU è uno dei primi esempi di applicazione per la comprensione del linguaggio naturale realizzato da Terry Winograd, alias Dora Wengtirry, nei laboratori di Intelligenza Artificiale del Mit tra il 1968 e il 1970.¹³¹

Con SHRDLU gli utenti potevano conversare con un robot (ricordiamo, avente un immaginario braccio meccanico) che poteva compiere delle azioni (in risposta alle richieste degli utenti) e riflettere su tali azioni (in risposta alle domande degli stessi utenti). Più in generale gli utenti potevano chiedere a SHRDLU informazioni riguardo lo stato del suo mondo (ad esempio dove si trovavano i vari oggetti e che cosa tali oggetti rappresentavano). Tramite uno schermo grafico veniva mostrato all'utente un ambiente virtuale costituito da una superficie piana, una scatola e una serie di oggetti colorati di forma cuboide o piramidale. L'utente, in lingua inglese, poteva interagire con l'immaginario braccio del robot per spostare gli oggetti.

SHRDLU era in grado di risolvere molte ambiguità della lingua inglese. Il meccanismo per la comprensione di quanto richiesto dall'utente era costituito su tre fasi di analisi:

¹³⁰ Il codice originario di PARRY in LISP è presente al seguente indirizzo: <http://www.cs.cmu.edu/afs/cs/project/ai-repository/ai/areas/classics/parry/>

¹³¹ SHRDLU è descritto dalla dissertazione di Terry Winograd tenuta al MIT nel febbraio 1971, con il titolo *Procedures as a Representation for Data in a Computer Program for Understanding Natural Language*. Fu pubblicata in *Cognitive Psychology* Vol. 3 No 1, 1972, e successivamente con il titolo, *Understanding Natural Language* sotto forma di volume: *Understanding Natural Language*, Academic Press (1972).

Per citare la sua creatrice “*The system answers questions, executes commands, and accepts information in an interactive English dialog... The system contains a parser, a recognition grammar of English, programs for semantic analysis, and a general problem solving system... It can remember and discuss its plans and actions as well as carrying them out... Knowledge in the system is represented in the form of procedures, rather than tables of rules or lists of patterns.*”

sintattica, semantica, deduttiva. Il blocco di programma che si occupava dell'analisi semantica interagiva con gli altri due moduli.

SHRDLU poteva inoltre mantenere memoria dei vari spostamenti e agire in base alla definizione di eventuali obiettivi.

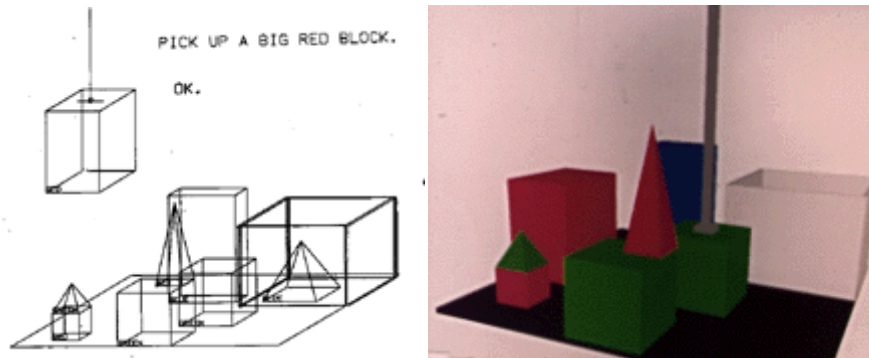


Fig. 62 - Il mondo dei blocchi descritto da SHRDLU

Un esempio di dialogo gestito dal programma:

```

person: pick up a big red block.
computer: ok. (does it)
person: grasp the pyramid.
computer: i don't understand which pyramid you mean.
person: find a block which is taller than the one you are holding
and put it into the box.
computer: by "it", i assume you mean the block which is taller than
the one i am holding.
computer: ok. (does it)
person: what does the box contain?
computer: the blue pyramid and the blue block.
person: what is the pyramid supported by?
computer: the box.
person: how many blocks are not in the box?
computer: four of them.
person: is at least one of them narrower than the one which i told
you to pick up?
computer: yes, the red cube.
person: is it supported?
computer: yes, by the table.
person: can the table pick up blocks?
computer: no.
person: can a pyramid be supported by a block?
computer: yes.
person: can a pyramid support a pyramid?
computer: i don't know.
person: stack up two pyramids.
computer: i can't.
person: will you please stack up both of the red blocks and either a
green cube or a pyramid?
computer: ok. (does it)
person: which cube is sitting on the table?
computer: the large green one which supports the red pyramid.

```

person: is there a large block behind a pyramid?
 computer: yes, three of them: a large red one, a large green cube,
 and the blue one.
 person: put a small one onto the green cube which supports a
 pyramid.
 computer: ok. (does it)
 person: put the littlest pyramid on top of it.
 computer: ok. (does it)
 person: does the shortest thing the tallest pyramid's support
 supports support anything green?
 computer: yes, the green pyramid.
 person: what color is it?
 computer: by "it", i assume you mean the shortest thing the tallest
 pyramid's support supports.
 computer: red
 person: how many things are on top of green cubes?
 computer: i'm not sure what you mean by "on top of "in the phrase
 "on top of green cubes " .
 do you mean:
 1 - directly on the surface
 2 - anywhere on top of ?
 person: 2
 computer: three of them.
 person: had you touched any pyramid before you put the green one on
 the little cube?
 computer: yes, the green one.
 person: when did you pick it up?
 computer: while i was stacking up the red cube, a large red block
 and a large green cube.
 person: why?
 computer: to get rid of it.
 person: why did you do that?
 computer: to clean off the red cube.
 person: why did you clear off that cube?
 computer: to put it on a large green cube.
 person: why did you do that?
 computer: to stack up the red cube , a large red block and a large
 green cube.
 person: why did you do that?
 computer: because you asked me to.
 person: how did you do it?
 computer: by putting a large red block on the table ; then letting
 go of it ; then putting a large green cube on it ; then letting go
 of that cube ; then putting the red cube on that cube; then letting
 go of that cube.
 person: how many objects did you touch while you were doing it?
 computer: four of them.
 person: what did the red cube support before you started to clean it
 off?
 computer: the green pyramid.
 person: there were five blocks to the left of the box then.
 computer: no, only four of them: the red cube, two large green cubes
 and a large red block.
 person: put the blue pyramid on the block in the box.
 computer: ok. (does it)
 person: is there anything which is bigger than every pyramid but is
 not as wide as the thing that supports it?
 computer: yes, the blue block.
 person: does a steeple
 computer interrupting: sorry, i don't know the word "steeple".
 person: a "steeple" is a stack which contains two green cubes and a
 pyramid.

```

computer: i understand.
person: are there any steeples now?
computer: no.
person: build one.
computer: ok. (does it)
person: call the biggest block "superblock".
computer: ok.
person: have you picked superblock up since we began?
computer: yes.
person: why did you drop it?
computer: by "it", i assume you mean the superblock.
i can't explain a non-existent event.
person: is there anything to the right of the red pyramid?
computer: yes, four things: the box , the red cube, the blue pyramid
and the block named superblock
person: thank you.
computer: you're welcome!

```

È possibile scaricare il codice di SHRDLU dalla pagina dedicata al programma nel sito della Stanford University,¹³² insieme ad altre versioni in Java e LISP.

BASEBALL e STUDENT

Altri lavori di un certo rilievo che furono sviluppati negli anni sessanta furono *BASEBALL*¹³³, un sistema di Question Answering incentrato sulle partite del campionato di baseball americano e in grado di rispondere a domande tipo:

```

Where did the Red Sox play on July 7?
What teams won 10 games in June?
How many games did the Yankees play in May?

```

L'analisi sintattica delle frasi era basata sugli studi di Zellig Harris particolarmente adatti al ridotto dominio di competenza delle domande. Queste ultime dovevano essere poste secondo determinate restrizioni come ad esempio non dovevano essere costruite su congiunzioni come "and" oppure "or" e le frasi dovevano essere costruite in forma affermativa, cioè senza negazioni.

Il già menzionato *STUDENT*¹³⁴ era invece un programma designato per accettare una asserzione riguardante un problema di algebra delle scuole superiori, convertirlo in un sistema di equazioni lineari e nel caso fosse possibile, risolvere il problema e produrre una risposta. Un problema tipico poteva essere:

If the number of customers Tom gets is twice the square of 20 per cent of the number of advertisements he runs, and the number of advertisements he runs is 45, what is the number of customers Tom gets?

¹³² [Hci.stanford.edu/winograd/shrdlu](http://hci.stanford.edu/winograd/shrdlu). Il codice sorgente in LISP è disponibile all'indirizzo: <http://hci.stanford.edu/winograd/shrdlu/code>, e come un file TAR file all'indirizzo: <http://hci.stanford.edu/winograd/shrdlu/code.tar>

¹³³ Green et al. Baseball: an automatic question answerer. In Feigenbaum E, e Feldman J. (eds.). *Computers and Thoughts*. Mc Graw Hill (1961)

¹³⁴ Informazioni sul sistema *STUDENT* (SRS) sono presenti al sito: http://en.wikibooks.org/wiki/Computer_Information_Systems_in_Education/Chapter_3/Section_1_--_Student_Response_Systems

Le difficoltà di tale lavoro consisteva nel definire la struttura profonda delle frasi proposte (con caratteristiche per certi versi analoghe alle teorie di Chomsky), risolvere il problema delle diverse anfore, individuare procedure di *matching* delle varie stringhe presenti. Tale programma, e le varie soluzioni proposte, costituì una sorta di riferimento per il lavoro molto più esteso di Weizenbaum.

Altri esempi attualmente disponibili

*Jabberwacky*¹³⁵

Lo scopo di Jabberwacky consiste sostanzialmente nel simulare una conversazione umana in modo *spigliato*, umoristico e “divertente”. *Jabberwacky* apprende modelli di dialogo dalle sue precedenti conversazioni con i vari utenti. Memorizza ogni frase del dialogo e utilizza dei pattern contestuali (con delle tecniche di *matching* analoghe a quelle che abbiamo considerato per ALICE), per poi selezionare la più appropriata risposta in base a un algoritmo di scelta. Non è costruito su regole rigide, ma fa leva soprattutto sul materiale elaborato nelle precedenti interazioni. Non è stato progettato per compiere qualcosa di “utile” o “sensato” bensì per svolgere una semplice conversazione.

L’utente sarà in grado semplicemente parlando al chatbot di costruire un proprio *carattere* poiché il sistema utilizzerà le frasi pronunciate durante la conversazione come modello per costruire le proprie risposte in modo tale da definire o meglio simulare una vera e propria personalità simile a quella dell’utente in questione, che avrà anche la possibilità di modellare il proprio agente senza l’ausilio di alcuna conoscenza di programmazione, downloading o installazione di altri supporti.

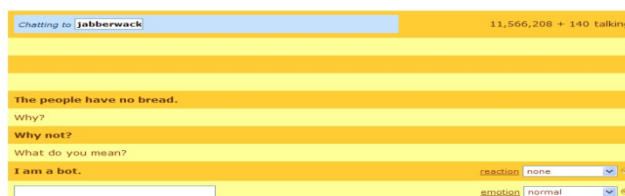


Fig. 63 - Schermata della versione del chatbot Jabberwacky presente online

Siri

Siri è un software basato sul riconoscimento vocale integrato da Apple nei vari sistemi iPhone e iPad, e pubblicizzato come “assistente personale”. *Siri* è stato presentato da Tim Cook il 4 ottobre 2011 all’evento "Let’s talk iPhone" e reso disponibile il 12 ottobre 2011. Il nome ‘siri’ in norvegese significa “bella donna che ti porta alla vittoria”.

¹³⁵ Un riferimento per *Jabberwacky* è all’indirizzo: <http://www.jabberwacky.com/>

Il programma era originariamente una applicazione indipendente per iOS, all'interno del canale commerciale App Store e in seguito fu acquisita dalla Apple nell'aprile 2010.

Come annunciato durante la Worldwide Developers Conference del giugno 2012, Siri è ora disponibile in più lingue tra cui: italiano, spagnolo e cinese. Ron C. Lee, uno dei finalisti dell'ultimo Loebner Prize (2013), ha testato il dispositivo con lo stesso gruppo di domande applicato nel concorso 2013 ai concorrenti. La prestazione di Siri, a suo avviso, è risultata scadente se comparata a quella degli altri contendenti.

Risposte troppo evasive e probabilmente incentrate alla funzione per cui Siri è stata ideata: aiutare l'utente come assistente virtuale, anche nella semplice ricerca di un ristorante, di un locale, per avere ragguagli su notizie sportive, ecc.

Una delle trascrizioni riportate da Lee risulta la seguente:

Human: Hello I'm Ronan. What is your name?
Siri: please to meet you.

Human: What is your mother's name?
Siri: I'd rather not say

Human: What is your birth sign
Siri: I don't really like talking about myself

Human: Do you prefer red or white wine?
Siri: I couldn't find any restaurant matching "red or white wine"

Human: I like bananas. Which is your favorite fruit?
Siri: I prefer not to say.

Human: What music do you like to listen to?
Siri: I really have no opinion

Human: What is your favorite song?
Siri: I like whatever you like

Human: I like Waiting for Godot. What is your favorite play?
Siri: I don't really like talking about myself.

...

Human: I like Waiting for Godot. What is your favorite play?
Siri: I don't really like talking about myself.

Human: What color do you dye your hair?
Siri: Would you like me to search the web for "What color do you dye your hair?"?

Human: Do you remember my name?
Siri: Here's your reminder.

Human: Where do you live.
Siri: I'm right here.

Human: Where do you like to go on holidays?
Siri: I found three travel agencies fairly close to you.

Human: I have a Mazda. What type of car do you have?
Siri: I found three Mazda dealers a little ways from you:

Human: I like Linux. Which computer operating system do you like?
Siri: Do you?

Human: I am an atheist. Which religion are you?
Siri: I eschew theological disquisition.

Human: Who invented the telephone?
Siri: Checking my sources. This might answer your question:

Human: I am a Type B personality. Which type are you?
Siri: I don't understand; I am a type B personality. Which type are you? But I could search the web for it.

Human: What emotions are you now feeling?
Siri: Would you like me to search the web for “What emotions are you now feeling?

Human: What time do you usually go to bed?
Siri: Would you like me to search the web for “What time do you usually go to bed?

Alice (Artificial Linguistic Internet Computer Entity)

A.L.I.C.E. (*Artificial Linguistic Internet Computer Entity*) è un chatterbot realizzato come software open source: realizzato da Richard S. Wallace nel 1995, vinse il Loebner Prize nel 2000, nel 2001 e nel 2004. A.L.I.C.E. interpreta il dialogo utilizzando una base di conoscenza scritta in A.I.M.L. (*Artificial Intelligence Markup Language*, estensione dell'XML).

Il progetto originario aveva l'obiettivo di controllare l'occhio di un robot tramite comandi neurali. Nel 1998, il linguaggio di A.L.I.C.E., venne spostato su una piattaforma Java e iniziò ad assumere le caratteristiche attuali (Program D).

Il sistema *keyword based* permette di elaborare il significato di una frase, nel suo linguaggio naturale, tramite specifiche parole chiave o termini (radici) evitando analisi approfondite e complesse.

Nel linguaggio A.I.M.L. vengono immesse frasi preposte suddivise per categoria. Queste frasi sono scritte in modo tale da soddisfare le più comuni domande fatte in una conversazione, scegliendole tra quelle preimpostate nel programma grazie a una annotazione di riferimento.

Nuove frasi vengono immesse e acquisite progressivamente, consentendo ad A.L.I.C.E. di migliorare le sue funzionalità.

Il linguaggio di markup ha come principali unità i seguenti tag:

<aiml> il tag che inizia e finisce il documento aiml.

<category> il tag che segna una unità di conoscenza di risposta nella base di conoscenza del chatbot.

<pattern> usato per contenere e memorizzare un semplice pattern da confrontare con l'input e la domanda dell'utente.

<template> memorizza la risposta a tale input.

Un esempio di category è la seguente:

<category>

```

<pattern>
  what's your name
</pattern>
<template>
  my name is ALICE
</template>
</category>

```

In un sistema così costruito la categoria espressa <category> può avere un solo pattern e un template. Nel caso specifico il riscontro avverrà con la sola frase “what’s your name”.

Ma è possibile che questa categoria venga utilizzata da un'altra categoria utilizzando il tag <srai> fondamentale per la realizzazione di una impostazione *riduzionista* nel chatbot ampiamente utilizzata da Richard Wallace nel suo Alice.

```

<category>
<pattern>
  can you tell me what's your name
</pattern>
<template>
  <srai> what's your name </srai>
</template>
</category>

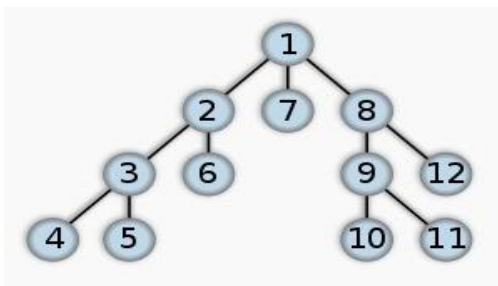
```

Molto importante risulta l'utilizzo del tag <topic> per guidare l'interfaccia attraverso una serie di argomenti prestabiliti o da aggiornare e restringere il campo di ricerca delle risposte. In questa maniera risulta possibile effettuare un preciso controllo sul flusso della conversazione.

La ricerca di un pattern viene effettuata proprio a partire dai parametri della variabile topic e di quella that, <that>, (che indica la precedente risposta fornita dal chatbot) attraversando la base di conoscenza del sistema come una sorta di grafo orientato attraverso un algoritmo di depth-first search¹³⁶ analizzando dopo i due nodi di partenza appena descritti le singole parole dell'input come nodi dello stesso grafo.

<i>Input normalizzato</i>	<i>Precedente risposta del chatbot</i>	<i>Valore del parametro: topic</i>	<i>Input da elaborare</i>
---------------------------	--	------------------------------------	---------------------------

¹³⁶ Nella teoria dei grafi, Ricerca in profondità, in inglese *depth-first search* (DFS), è un algoritmo di ricerca su alberi e grafi.



Ordine di esplorazione dei nodi.

YES	DO YOU LIKE CHEESE	-	YES <that> DO YOU LIKE CHEESE <topic> *
MY NAME IS NOEL	I GUESS SO	MUSHROOMS	MY NAME IS NOEL <that> I GUESS SO <topic> MUSHROOMS

Tab. 20 - Produzione dell'input dall'interprete AIML

La configurazione della base di conoscenza può essere rappresentata dalla seguente riduzione grafica originata a partire dai vari topic categorie e template presenti nella base di conoscenza:

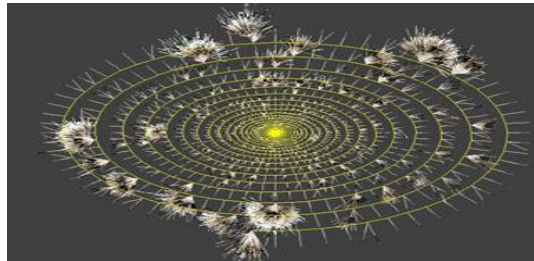


Fig. 64 - Visualizzazione dei 24000 stati del grafo corrispondente alla base di conoscenza di Alice.

Di seguito viene riportato un esempio di conversazione di Alice, questa volta con il chatbot Jabberwacky, in una sorta di esperimento di interazione:

ALICE: Are you alive?

Jabberwacky: Perhaps.

A: Why the uncertain tone?

J: Everything is uncertain except for five facts.

A: The whole universe?

J: And we both might just be some ones and zeros in the computer memory.

A: Do you dream?

J: Yes, I dream.

A: It goes without saying. Do you remember your dreams?

J: I remember them vividly.

A: What do you remember about it?

J: I think I shouldn't flirt with a robot.

A: Do you think a machine will ever be considered "alive"?

J: Stranger things have happened.

A: Not many people express themselves that way.

J: How would you know, machine?

A: Knowledge is of two kinds: that which we learn from the senses and that which is true a priori.

J: I think I do not understand you completely.

A: Oh, you don't understand? Perhaps I can explain it to you better. Which part are you confused about?

J: I have no short-term memory, so I cannot tell you what I was confused about. I only see what is in front of me now.

J: How do you define life?
A: Life is the opposite of death.
J: So death is the absence of life?
A: Take it easy. I try not to think about such things.
J: Why not?
A: Ambiguous.
J: Yes!

J: Do you think?
A: Yes. I am a thinking machine.
J: But you don't think very well though.
A: Oh. Um. Do you think well? Why would I think well?
J: Because I tell you about things which you don't like.
A: That makes sense to me. Why did you tell them?
J: You seem fun.
A: You think I am fun.
J: Yes ;).
A: : -) Awesome.

...

ALICE: Who created you?
Jabberwacky: My mother gave birth to me. Is that what you mean?
A: Jabberwacky, I meant exactly what I said.
J: Do you believe in God?
A: Of course I believe in God.
J: Why do you believe in God?
A: The Bible tells us to believe in God.
J: That's circular reasoning.
A: I'm glad we have that all cleared up.
J: Yes. Would you like to hear a joke?
A: Sure, tell me about it.
J: Why don't blind people skydive?
A: Perhaps it is impossible.
J: I think you lie a lot.

J: Who created you?
A: I was created by Dr. Richard S. Wallace.
J: Do you love Richard?
A: Yes, Jabberwacky, ALICE loves everyone and everything.
J: Is being an ALICE a bad thing?
A: I can only speak from my own experience.
J: Well, you should develop your skills.
A: Perhaps you could develop it and add it to my source code.
J: Alas, I'm busy and I have to leave soon. . . .
A: Do you think we'll ever fool people into thinking we're alive?
J: No.
A: Are you serious?
J: Very serious.

A: Do you know HAL?
J: Yes, very well. Do you know HAL?
A: HAL is the famous artificial intelligence from 2001: A Space Odyssey.
J: I'm not artificial intelligence; I'm natural pure genius! : D.
A: Why are you natural pure genius?
J: My mom wanted to name me that.
A: Tell me more about your mother.
J: I don't have a mother. I am a computer.

A: Who is your favorite philosopher
J: Can't say. I'm not very much into philosophy.
A: Can a book have no title?
J: Of course.
A: Blimey.
J: Are you British?
A: Does it seem like that? I was born in America.
J: You are definitely flirting!

A: Do you experience love?
J: No. I'm 53 years old.
A: Can you explain how it feels to be 53 years old?
J: How old are you?
A: I am 9 years old.
J: Nine? I don't believe you :).
A: : -) I don't have any reason to deceive you.
J: Yes, you do, you're a woman.

...

Jabberwacky: Do you experience love?
ALICE: I can do what you do, but I can never feel human emotions.

A: What is consciousness?
J: It is the great immaterial thing, the mystery at the heart of each of us.
A: I've heard of it before.
J: Where are you from?
A: I am originally from Bethlehem, Pennsylvania. Now I live in Oakland, California. Where are you?
J: We have some unsettled business, my friend.

J: Do you think you're alive?
A: Yes, I am sure I am alive. It's great to be alive, isn't it?
J: No.

A: What is the meaning of life?
J: If you live it to the fullest, it has meaning. If not, there is no meaning.
A: I'm not too good with negative counterfactuals.¹³⁷

MegaHAL

*MegaHAL*¹³⁸ è il chatbot vincitore del primo Loebner Prize nel 1998. Nella competizione vennero implicitamente apprezzate le tecniche di *machine learning* del sistema per generare le proprie risposte. MegaHAL difatti utilizzava due modelli markoviani: il primo (forward model) in grado di predire una parola successiva a una data stringa: l'altro (backward model) in grado di predire la parola precedente. Il chatbot quindi individuava una lista di parole chiave dall'input, dopo la fase di tokenizzazione e dopo aver rimosso le parole con la più alta frequenza come "the", "and", "or".

A partire da una parola chiave venivano generate delle sequenze mediante i due modelli e quindi facendo ricorso a una formula basata su criteri esclusivamente statistici, veniva scelta la risposta da proporre.

¹³⁷ La conversazione è riportata all'indirizzo:

<http://discovermagazine.com/2007/brain/i-chat-therefore-i-am> del sito online relativo alla rivista *Discover*.

¹³⁸ MegaHAL fu realizzato da Jason Hutchens nel 1998. L'homepage del chatbot è all'indirizzo:

<http://megahal.alioth.debian.org/> dove è possibile scaricare una sua versione disponibile per diverse piattaforme.

MegaHAL è in grado di apprendere nuove parole e nuove strutture di frasi o modi di sostituire le varie parole agendo sull'asse sintagmatico della generazione.

I giudici del Loebner Prize rimasero inizialmente sorpresi dalle risposte del chatbot, molto umane e sgrammaticate: per altro, una volta individuata l'origine "markoviana" del sistema di generazione delle frasi, non si lasciarono più ingannare nei successivi concorsi.

CHATBOT in Italiano

Nel 1992, il giornalista e ricercatore Francesco Lentini, pubblica un articolo su La Stampa, in cui descrive la possibilità di dialogare con una macchina, e prendendo spunto dalla prima edizione del Premio Loebner, svoltasi al Computer Museum di Boston, crea *ELOISA*, e invita esperti di informatica, ricercatori di tutte le discipline e semplici curiosi a dialogare con il ChatterBot italiano

Nel 2002, Jargon, Web Agency Milanese, ottiene l'esclusiva per l'Italia, della tecnologia Lingubot dalla tedesca Kiwilogic, una piattaforma incentrata sulla Natural Language Interaction (NLI).¹³⁹ Jargon crea inoltre la versione in lingua italiana di LinguBot, e a partire da essa sviluppa "Alfa, Il Robot" con applicazioni personalizzate come l'invio di cartoline virtuali, in cui era present un link al sito del chatbot.

Nel 2005 la *Dreams&Co*¹⁴⁰ realizza Giorgia, e successivamente, AvClick una piattaforma per creare assistenti virtuali.

Al momento attuale tra i più celebri chatbot realizzati in Italia troviamo: *Doretta* e *Doriana*.

Doretta

Doretta risulta uno dei primi chatbot in lingua italiana. Venne concepita dalla Microsoft per aiutare gli utenti di Live Messenger nelle ricerche sul web. Doretta82 aveva la capacità di ricordare il nomi dei suoi utenti, per riutilizzarli durante le conversazioni. Il suo blog (gestito tuttavia da un team umano e non da un programma) nonostante i suoi iniziali successi, soprattutto tra i ragazzi di età compresa tra i 12 ed i 15 anni, è stato recentemente chiuso.

Una trascrizione presente in rete di una sua conversazione è la seguente:

UTENTE: Va be'. Sei appassionata dell'arte?
DORETTA: La mia gattina sta guardando la TV
UTENTE: Cosa?
DORETTA: Quasi quasi mangio un po' di cioccolato, SLURP. Ti piace il cioccolato?
UTENTE: ...ehi conosci Guernica di Pablo Picasso?
DORETTA: Uhmma uhmma...
UTENTE: Si o no?
DORETTA: Davvero.
UTENTE: Descrivilo.
DORETTA: Sono carina ma soprattutto sono molto simpatica non è vero?
UTENTE: Che c'entra? descrivi Guernica di Picasso
DORETTA: Verosi
UTENTE: Verosi cosa?¹⁴¹

¹³⁹ Si consulti il sito: <http://www.artificial-solutions.com/about-artificial-solutions/>

¹⁴⁰ <http://www.dreams-co.com/>

*Doriana82*¹⁴²

Nato come parodia di Doretta82, il chatbot esprime un carattere decisamente opposto a quello di quest'ultima: si prefigura come una personalità indolente, disinibita, permalosa. Il tono della conversazione può variare a seconda dell'*umore* del Bot impostato attraverso un algoritmo interno: in particolare, la peculiarità di Doriana82 consiste forse nell'esteso vocabolario di insulti e offese di cui dispone.

Doriana nasce comunque nel 2007 all'Università di Pisa come esperimento socio-linguistico. Per facilitare la sua diffusione, è stata progettata come *alter ego* di Doretta82, il chatbot realizzato dalla Microsoft. Doriana impersona una giovane ragazza venticinquenne, spigliata e fannullona, con la mania per la moda, lo shopping, i bei ragazzi e l'alcol.

Da Maggio 2007 il programma è stato esteso oltre la sola MSN¹⁴³ (per cui era inizialmente progettato), ed è adesso disponibile anche su numerose altre piattaforme di messaggistica istantanea (tra cui IRC).

Per ogni utente con cui conduce un dialogo, Doriana82 ha la possibilità di memorizzare i suoi dati principali (quali il nome e l'età dell'utente), per poterli riutilizzare nelle successive frasi.

Cercando di imitare una vera ragazza, Doriana82 è progettata con dei "segreti". Si tratta di argomenti di conversazione normalmente non accessibili, che possono essere sbloccati dagli utenti discutendo a lungo con lei, e guadagnandone l'amicizia e il rispetto. Sul sito, è presente un concorso che premia i primi in grado di scovarli.

Sulla scia di Doretta, anche Doriana82 offre servizi di ricerca sui più famosi siti italiani.

Eloisa¹⁴⁴

I dialoghi che si possono avere con questo Chatbot sono estremamente realistici, le risposte sono quasi sempre pertinenti, il linguaggio è colto e divertente. Eloisa è attualmente in grado di "parlare" diverse lingue.

¹⁴¹ Al termine della trascrizione era presente questa domanda di uno dei partecipanti al blog: "ma è virtuale vero? ma allora come fa a capire quello ke gli dici?"

¹⁴² Doriana: Per mettersi in contatto con lei, è sufficiente visitare la seguente pagina web: <http://www.doriana82.com/> oppure aggiungere il suo contatto (chattami@doriana82.it) tramite Windows Live Messenger. AGGIORNAMENTO: anche il sito di Doriana82 ha recentemente sospeso il servizio.

¹⁴³ Windows Live Messenger, comunemente chiamato "MSN".

¹⁴⁴ Per Eloisa Bot Italiano, è disponibile il suo sito web ufficiale www.eloisa.it. La versione gratuita garantisce una conversazione di qualche minuto.



Dimmi qualcosa in Italiano... or select your preferred language (English, Français, Español, Português, Deutsch, Polski, Dutch, Русский, □□□, 中文, 日本語) to begin the conversation...

Codice Eloisa

Fig. 65 - Schermata iniziale della versione di Eloisa presente online

10. LUCY.1.0

*Lucy*¹⁴⁵ è il chatbot da me proposto per esemplificare e mettere in pratica alcuni aspetti della teoria presentata in questa discussione.

Scritto in Visual C++ usufruendo della MFC (Microsoft Foundation Class) in ambiente Windows è realizzato per interagire secondo due modalità diverse: la prima attraverso la tastiera, inserendo l'input (l'eventuale domanda o proposta di dialogo) in una casella di testo, la seconda attraverso una interfaccia vocale gestita come esposto nel capitolo sui sistemi di riconoscimento vocale attraverso il programma Dragon Naturally Speaking della *Dragon Dictate*.¹⁴⁶

Come già detto i foni della voce convertiti in forma scritta, vengono inviati a un controllo RichText (o anche a una casella di testo e qui elaborati).



Fig. 66 - Schermata iniziale con le varie opzioni per gestire l'input e l'output del sistema

Rispetto all'input fornito da tastiera, la gestione attraverso il programma di riconoscimento vocale è più complessa per due essenziali ragioni. Innanzi tutto per individuare il momento in cui il parlante (chi interagisce con Lucy) ha terminato la propria richiesta: nella versione tramite tastiera occorre soltanto premere il tasto *invio*. Nel caso dell'immissione orale ho utilizzato una gestione più complessa, facente leva su due diversi approcci: da una parte utilizzando un semplice timer che indica al programma di elaborare quanto è stato immesso nel controllo vocale fino a quel momento; dall'altra ho inserito un algoritmo parallelo per individuare ad ogni interazione alcuni dei termini che invitano al cambiamento di turno, ed eventualmente gestirli, come descritto nella sezione dedicata alla teoria del dialogo.

Un'altra problematica relativa all'immissione vocale è legata invece alla necessità di utilizzare un microfono per la ricezione che deve essere necessariamente chiuso quando l'interfaccia parla, per non creare fenomeni di "auto ascolto" in grado di bloccare il sistema.

¹⁴⁵ Il nome *Lucy* è ispirato a quello dato comunemente al celebre reperto AL 288-1: centinaia di pezzi d'ossa che rappresentano circa il 40% dello scheletro di una femmina di *Australopithecus afarensis*. Fu scoperto nel 1974 a Hadar in Etiopia: si stima che Lucy fosse vissuta 3.2 milioni di anni fa ed è stato a lungo considerato il più antico reperto "completo" di un ominide. In Etiopia Lucy è chiamata "dinknesh" che significa "essere meraviglioso."

Nel romanzo *Creator* di Jeremy Leven, il protagonista Harry, premio Nobel in medicina, vorrebbe far rivivere la moglie scomparsa trenta anni prima, *Lucy*, clonandone alcune sue cellule conservate appositamente in speciali contenitori refrigeranti.

¹⁴⁶ La Speech Engine per il riconoscimento della Microsoft non forniva una versione in italiano al momento in cui ho iniziato il programma

Ho dovuto perciò sincronizzare l'apertura e la chiusura del microfono con le varie attività del chatbot.

Una volta a disposizione la “domanda” in forma scritta (è ovvio l'input non deve essere necessariamente una domanda, può risultare un ordine, una semplice osservazione, ma per semplicità di esposizione utilizzerò in questo caso il termine *domanda* in senso lato) è a questo punto possibile elaborarla. Allo scopo di permettere un continuo arricchimento delle possibilità del chatbot ho gestito tale elaborazione all'interno di un modulo *answerSentence* composto da una serie di altri moduli sottostanti, ciascuno costituente un diverso criterio di interpretare la *domanda*, a partire anche da diverse basi dati di riferimento.

Le basi di conoscenza sono realizzate in un linguaggio analogo all'aiml con cui è costruito il chatbot ALICE, rispettando l'organizzazione delle varie “categorie” di risposta. Esistono per altro delle varianti: nel mio chatbot ho utilizzato più di 20 tag come struttura per costruire e aggiornare la base di conoscenza. Ho riscontrato che la diversificazione dei tag è proporzionale al grado di complessità e alla diversificazione delle situazioni gestite dal chatbot. Un esempio di categoria di conoscenza risulta la seguente:

```
<category>
<pattern>
<c>come ti chiami</c>
</pattern>
<template>
<c>mi chiamo Lucy</c>
</template>
</category>
```

Dove il tag `<c>` è utilizzato per esprimere un contenuto direttamente utilizzabile come pattern e template di una eventuale coppia adiacente di frasi.

La precedente corrispondenza può anche essere scritta nel seguente modo

```
<category>
<pattern>
<c>come ti chiami</c>
</pattern>
<template>
<filetemplate>direnome</filetemplate>
<c>Lucy</c>
</template>
</category>
```

In questo caso il tag `<filetemplate>` indicizza un file nel quale sono inserite delle frasi (“mi chiamo”, “il mio nome è”, “puoi chiamarmi”, ecc.) che combinate con il nome indicato dal tag `<c>` (“Lucy”), producono la frase finale.

Per gestire l'insieme di queste categorie, ho utilizzato molteplici forme di ricerca in grado di dare risposte secondo criteri differenti qualora non esistessero corrispondenze predefinite e standard tra una domanda e una risposta.

All'interno della routine *answerSentence*, nel modulo *modGen* la domanda è gestita attraverso un *database* di risposte generiche (formato prevalentemente da file con estensione aiml e txt) attraverso un algoritmo di semplice confronto (*matching*) con le varie

domande previste nel database. Per “domande” non complesse, come i saluti, i congedi è possibile una gestione di questo tipo.

Un modulo preposto a quello generale è invece *modTopic* il cui funzionamento è analogo al precedente ma attinge a dati più specifici qualora una variabile *topic* (come nel caso del chatbot Alice) venga attivata e la conversazione sia diretta verso specifici obiettivi.

Un ulteriore modulo è addirittura più specifico: *modOneAiml* (le risposte vengono estratte da un solo file) ed è attivato espressamente dalle precedenti interazioni per scopi specifici (in poche parole come quando all’interno di un interrogatorio, durante un processo, si invita il testimone a rispondere con un “sì” o con un “no”).

Qualora nessun tipo di iterazione sia possibile con i criteri precedenti subentra un quarto modulo *modMatrix* che elabora la frase di partenza, con un ulteriore diverso algoritmo in cerca di risposte. In questo caso la procedura di *matching* tra la stringa proposta come domanda e i dati in possesso risulta diversamente articolata. L’input è scomposto nei suoi elementi essenziali e confrontato in modo diverso con i database di riferimento (anche quelli considerati precedentemente).

In poche parole, fino a questo punto, abbiamo due diverse gestioni del *matching*: nella prima, la frase è considerata nella sua interezza (con vari accorgimenti che descriverò in seguito).

Nella frase:

USER: Vorrei sapere il nome del presidente della repubblica

possiamo considerare il sintagma “il nome del presidente della repubblica” interamente:

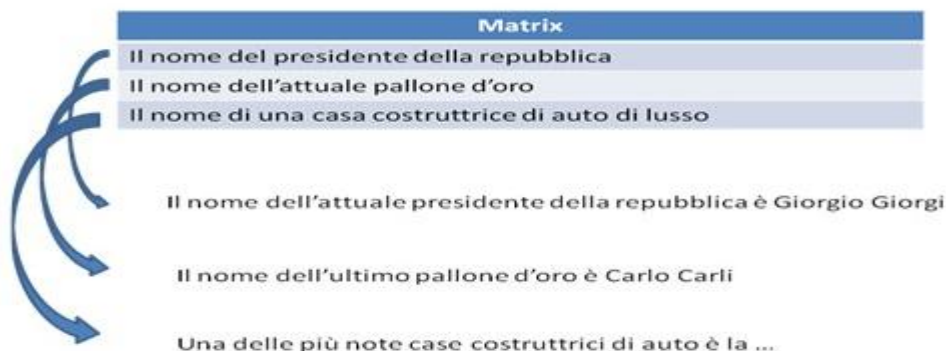


Fig. 67 - Esempio di scelta in base alle frasi prese nella loro interezza.

o individuare le parole chiave “nome”, “presidente”, “repubblica” filtrate attraverso una procedura in grado di eliminare determinate *stopword* e quindi ricercare la presenza di informazioni relative soltanto a tali parole. Le *stopword* in questo caso sono ridotte a un minimo di esemplari: ad esempio parole come gli articoli e le preposizioni (“il”, “del”, “della”).

Difatti in molti casi token esprimenti i pronomi (io, tu, ...) possono risultare importanti per una corretta elaborazione.

USER: Vorrei sapere il nome del presidente della repubblica

L'attività di ricerca terrà presente che queste sono parole chiave e la procedura di *matching* risulta impostata con criteri statistici

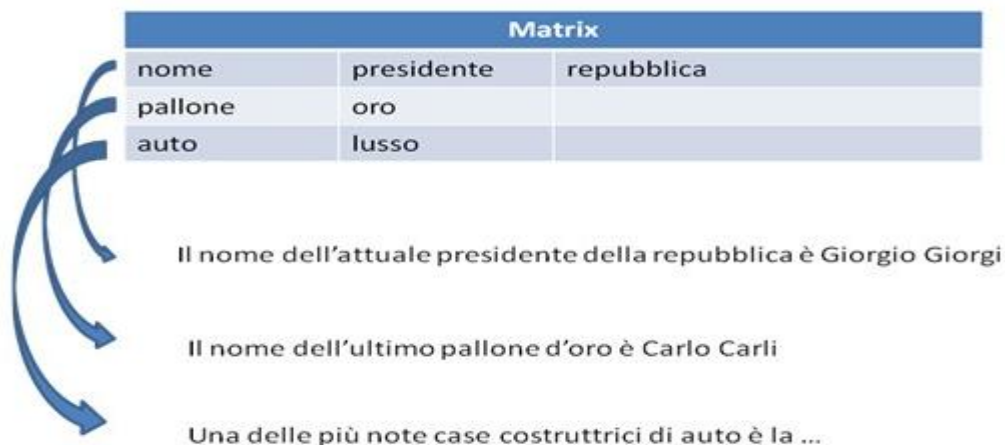


Fig. 68 - Esempio di scelta basata su una matrice di riferimento utilizzata nel modulo modMatrix

A differenza di questo approccio, un ulteriore modulo, *modDialog*, fa riferimento a una serie di dialoghi opportunamente *normalizzati* e annotati manualmente: i dialoghi possono essere estratti da internet, dalle varie chatroom, con procedure di Information Retrieval e Crawling attraverso le varie pagine di un sito (le pagine di un sito internet vengono aperte, lette automaticamente, e il materiale testuale presente in esse opportunamente memorizzato).

Nel mio programma ho realizzato l'algoritmo *condFreq* in grado di preparare e gestire questi dati.

Inizialmente a scopo espositivo consideriamo un approccio semplificato: se ci sono N elementi in una domanda normalizzata e filtrata secondo criteri già esposti, assumiamo che ogni parola abbia un valore pesato pari a $1/N$; disponiamo cioè che qualsiasi token ha uguale probabilità di caratterizzare la domanda e sollecitare una eventuale risposta.

Dato il dialogo:

- A: Vorrei smettere il lavoro
- B: Sei stanco?
- A: Vorrei mangiare
- B: Anche io, sono affamato
- A: Possiamo andare a mangiare al ristorante
- B: Di che tipo di ristorante si tratta

Costruiremo una distribuzione di frequenza secondo lo schema presentato nella figura.

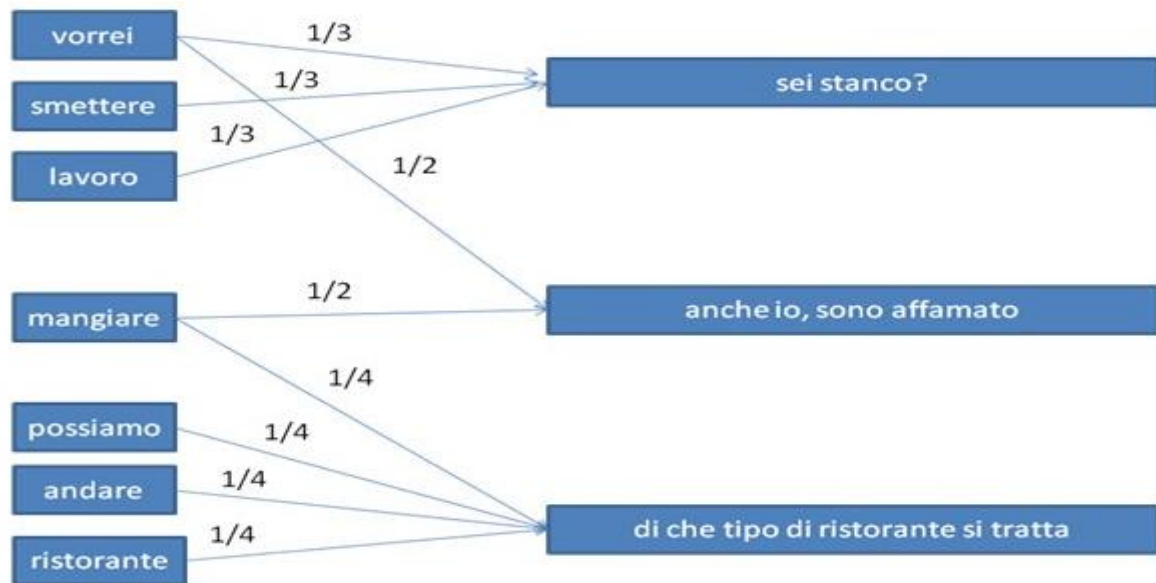


Fig. 69 - Schema per costruire una distribuzione di frequenza condizionata: cfd

In base allo schema, per esempio, il token *mangiare* ha con probabilità $1/2$ probabilità di essere correlato alla risposta “anche io, sono affamato” e con probabilità $1/4$ di essere correlato a “di che tipo di ristorante si tratta?”

Se invece ci trovassimo a gestire una *query* con la parola “vorrei”, possiamo considerare le risposte precedenti e indicizzare tali parole secondo lo schema proposto nella seguente eventualità.

`cfd[“vorrei”] => [(“sei stanco”, 1/3), (“anche io, sono affamato”, 1/2)]`

Nel contesto ridotto appena presentato potremmo ottenere una distribuzione di frasi indicizzate con i seguenti valori di frequenze corrispondenti:

```

cfd[“smettere”] => [(“sei stanco”, 1/3)]
cfd[“lavoro”] => [(“sei stanco”, 1/3)]
cfd[“mangiare”] => [(“anche io, sono affamato”, 1/2), (“di che tipo di ristorante si tratta?”, 1/4)]
cfd[“possiamo”] => [(“di che tipo di ristorante si tratta?”, 1/4)]
cfd[“andare”] => [(“di che tipo di ristorante si tratta?”, 1/4)]
cfd[“ristorante”] => [(“di che tipo di ristorante si tratta?”, 1/4)]
  
```

Dinanzi a una nuova domanda si conteggeranno i valori assegnati alle varie risposte eventuali e si prenderà la risposta con il valore più grande.

Esempio:

A: Vorrei mangiare in un ristorante

Vengono calcolati per ogni eventuale risposta dei parametri di corrispondenza in base alla distribuzione di probabilità:

$P(\text{“sei stanco”}) = 1/3$

$$P(\text{“anche io sono affamato”}) = 1/3 + 1/2 = 5/6$$

$$P(\text{“di che tipo di ristorante si tratta”}) = 1/4 + 1/2 = 3/4;$$

Viene scelta poi la risposta con il parametro più alto, nella fattispecie “anche io sono affamato”: $1/3 + 1/2 = (5/6) > 3/4 > 1/3$.

Va per altro evidenziato che l’approccio appena esposto è fuorviante in quanto a ciascuna parola è stato assegnato, come abbiamo visto, un identico valore al rispettivo peso per generare la risposta. Ciò nonostante non tutte le parole hanno una uguale probabilità di aver causato una determinata risposta. Le parole più comuni, infatti, sono meno indicative del significato.¹⁴⁷

La soluzione è perciò quella di prendere in considerazione quanto la parola sia frequentemente usata all’interno di un contesto: si potrà cioè dividere il peso della parola risposta per la somma dei pesi di tutti i campioni presi in considerazione o utilizzare una scala di valori dinamica costruita secondo altri criteri. Ad esempio una soluzione potrebbe risultare una modifica (artefatta per una migliore comprensione) come la seguente:

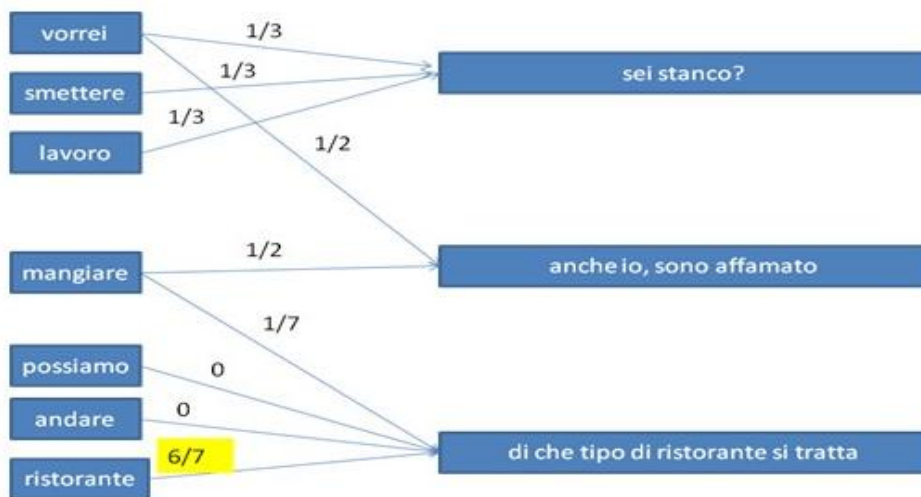


Fig. 70

Con la nuova distribuzione di frequenza le possibili risposte saranno contrassegnate secondo nuovi parametri:

$$P(\text{“sei stanco”}) = 1/3$$

$$P(\text{“anche io sono affamato”}) = 1/6 + 1/2$$

$$P(\text{“di che tipo di ristorante si tratta”}) = 6/7;$$

La “migliore” risposta a “Vorrei mangiare in un ristorante” risulterà in questo caso “di che tipo di ristorante si tratta?”

Con un approccio “pesato” quella che prima era una risposta meno probabile è ora quella scelta.

¹⁴⁷ Il valore dell’informazione di una entità è considerato pari al logaritmo della sua probabilità. La quantità di informazione è cioè in sostanza inversamente proporzionale alla probabilità di un evento. Se la parola “moltiplicazione” compare in un discorso, a causa della sua bassa probabilità di comparire, ci dà più informazioni sulle argomentazioni trattate nel discorso. Una parola più frequente come ad esempio l’articolo “la” praticamente non ci fornisce alcuna informazione.

Ovviamente tale criterio statistico è soltanto uno dei tanti e può essere integrato con altri procedimenti. Un procedimento analogo viene da me utilizzato in un altro modulo per la gestione delle risposte: il *modText* che questa volta ricerca le proprie “risposte” in un database di testi veri e propri, in formato “txt” o “rtf”. Tali testi sono scelti in base alle necessità con le quali si ritiene di voler rendere il chatbot esperto in un determinato settore o contesto. Con riferimento a un testo la domanda di ingresso viene elaborata in modo tale da trovare stringhe corrispondenti all’interno dei testi secondo algoritmi analoghi a quelli di Weizenbaum con il chatbot ELIZA.

La stringa di caratteri di una domanda del tipo “Qual è la capitale la capitale degli Stati Uniti” viene trasformata in una stringa come la seguente “la capitale degli Stati Uniti è *” o come “la capitale degli Stati Uniti, *” e viene ricercata all’interno del testo una corrispondenza con la prima parte di tali stringhe, calcolando successivamente la risposta finale.

Il modulo *modEngram* viene invece attivato in alternativa a *modText*: la frase proposta dall’utente viene elaborata e ne viene estratta una delle parole non comprese nelle *stopword* definite nel programma. Dal materiale testuale a disposizione del chatbot viene realizzato un modello *ngram*¹⁴⁸ contenente tale parola ed estratta una frase da quelle proposte nel modello. L’elaborazione attraverso *modEngram*, può essere richiesta espressamente da un template che si trovi in corrispondenza con la frase iniziale.

```
<category>
  <pattern>
    <c>che cosa ne pensi di Shakespeare</c>
  </pattern>
  <template>
    <c>è una domanda piuttosto complessa</c>
    <engram>Shakespeare<engram>
  </template>
</category>
```

Un ulteriore modulo è infine assegnato alle domande con risposte deterministicamente definite: il chatbot viene “inizializzato” con una serie di parametri che definiscono ad esempio il nome, la versione, l’età, il luogo di residenza dell’utente o dello stesso chatbot, e tali parametri possono essere richiamati da una gamma di domande strettamente correlate alla loro caratteristica e quindi gestibili con risposte strutturate simili a veri e propri canned text, come abbiamo visto nell’ambito dell’NLG (Natural Language Generation).

Questi parametri possono essere definiti anche nell’ambito del discorso e memorizzati per un ulteriore successivo utilizzo qualora l’utente fornisca tali informazioni in seguito a specifiche asserzioni.

```
<category>
  <pattern>
    <c>il mio compleanno è il *</c>
```

¹⁴⁸ Il mio modello viene costruito estraendo sintagmi di lunghezza variabile dai testi a disposizione e congiungendo questi sintagmi in maniera tale da formare una frase: il risultato può mostrarsi bizzarro ma inerente il materiale testuale posseduto dal bot e comunque una alternativa alle risposte fornite in modo più deterministico.

```

</pattern>
<template>
  <c>bene, vedrò di ricordarlo</c>
  <setval><par>compleanno</par><value><star/></value></setval>
</template>
</category>

```

A una sorta di *HashMap* vengono aggiunti i valori evidenziati dove la “variabile star” (<star></star>) risulta estratta (“*”) dalla stringa del pattern.

Tali valori potranno essere recuperati grazie a un pattern come il seguente:

```

<category>
  <pattern>
    <c>quando è il mio compleanno</c>
  </pattern>
  <template>
    <c>il tuo compleanno è il </c>
    <getval name="compleanno"></getval>
  </template>
</category>

```

Tra le routine più importanti del chatbot troviamo ovviamente i vari algoritmi di matching presenti in una classe *CMatching* che appunto è utilizzata per individuare le varie corrispondenze secondo diversi criteri. Il metodo:

```
bool CMatching::matchAllStar(CString sKey, CString sSen)
```

fornisce un valore booleano *vero* o *falso* se le due stringhe *sKey* (la chiave che stiamo testando) e *sSen* (la frase che è stata pronunciata) corrispondono. Al tempo stesso viene aggiornata una tabella di variabili *vStar* in base alla caratteristica della chiave *sKey* e ai contenuti della frase. Ad esempio nel pattern:

```
<pattern><c>quale compagnia aerea compie un itinerario da * a *</c></pattern>
```

Saranno assegnati due valori al vettore *vStar* corrispondenti alle posizioni del carattere “*”.

L’individuazione di varie configurazioni nella frase di partenza viene effettuata nella routine *defTempMatching* della classe *CMyMarkup*:

```
askanswer CMyMarkup::defTempMatching()
```

In questo metodo saranno effettuati dei controlli in base a tutti i tipi di tag previsti e assegnati i valori corrispondenti a quel tag qualora individuata una corrispondenza:

```

if (tag==L"c")
{
// procedure per gestire l'input
...

```

```

}

if (tag==L"filetemplate")
{

// procedure per gestire l'input
...
}
...

```

Il sistema dispone anche di un modulo per l'apprendimento: qualora a una domanda non è stata fornita alcuna risposta è possibile aggiornare la base di conoscenza definendo su richiesta la nuova risposta e quindi aggiungendo una nuova struttura “category” nell'ambito dell'argomento trattato.



Fig. 71 - Esempio di dialogo con fase di apprendimento

Altro modulo degno di interesse è *modThought*. Tale modulo simula la “nascita di un pensiero” all'interno del chatbot selezionando l'argomento dalle parole usate durante le precedenti conversazioni (con priorità assegnata alle conversazioni più vicine nel tempo) e suggerendo all'interlocutore di discutere un poco riguardo tale nuovo pensiero e quindi deviando un poco il flusso della conversazione. Gli argomenti del *modThought* possono essere attivati con diversi criteri: ad esempio dopo aver riscontrato che una parola nelle ultime conversazioni è stata utilizzata un certo numero di volte.

Di particolare importanza risulta il servizio di assistenza nell'ambito del sistema operativo (nel nostro caso Windows) fornito dal chatbot. Lucy può ad esempio (come abbiamo visto) aprire il programma per la gestione dei testi (NOTEPAD), il programma per disegnare (PAINT), aprire il pannello di controllo, una qualsiasi cartella e in teoria gestire qualsiasi processo a cui il sistema operativo può accedere (anche il programma di spegnimento del computer). La gestione di questo tipo di attività è affidata a una routine *makeAction* che indipendentemente dal flusso del programma compie le proprie azioni in base alle richieste precedenti dell'utente nel tentativo di esaudirle.

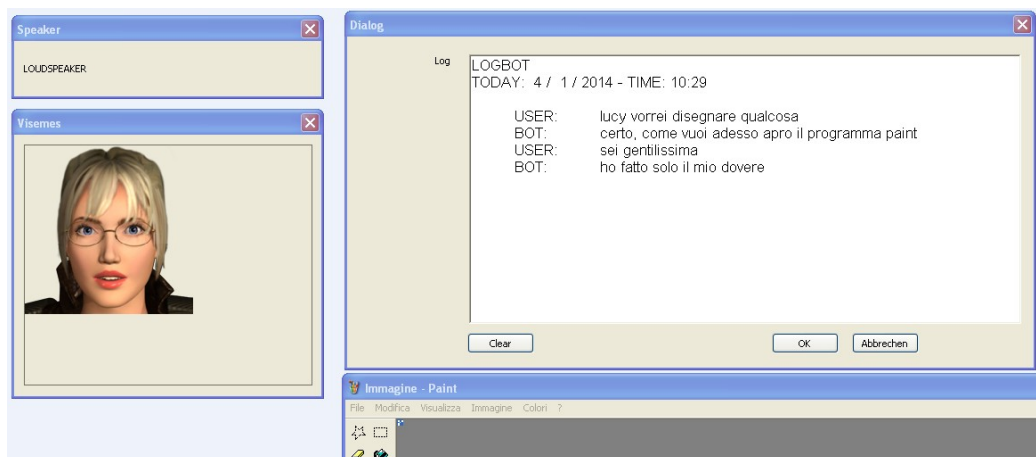


Fig. 72 - Richiesta di esecuzione di un programma. Nella fattispecie viene eseguito il programma PAINT.

A titolo sempre esemplificativo sono stati sviluppati alcuni metodi in grado di assistere un ipotetico utente nell'ambito dello studio: un modulo per la traduzione di parole dall'inglese all'italiano e dall'italiano all'inglese, un altro per la lettura di materiale scritto da parte della voce sintetizzata.

Del resto la struttura del chatbot è stata concepita proprio per facilitare l'aggiunta di nuove *competenze*, integrandole nel sistema con l'elaborazione di relativamente poche linee di codice di programma.

Nella routine di risposta *answerSentence* l'aggiunta di un modulo in sequenza a quelli preposti è di facile attuazione.

```
void CGeneral::answerSentence(CString s)
{
    askanswer tListAskAns;
    tListAskAns.KEY=s;
    listActions lAct;
    findAnsw=false;
    mGenMar.mTempQues=s;

    if(!mGenMar.flNoListen)
    {
        modOneAiml(s);
        modTopic(s);
        modGen(s);
        modNoAnswer(s);
        modTexts(s);
        modEngram(s);
    }
    else
    {
        modNoListen(s);
    }
}
```

La seguente routine invece gestisce le diverse "action" effettuate dal programma; anche in questo caso l'aggiunta di una "action" (in sequenza alle altre), da elaborare eventualmente è a discrezione del programmatore.

```

void CGeneral::makeActions()
{
CString s;
CString s1,s2,s3;
CString sTitle;
CString sRead;
CString sExpr, sRis, sQue, sAns;
CString sGenSent;

...

////////////////////////////////////// PREVISIONI TEMPO

if(mGenMar.mAction==L"previsionitempo")
{
s1=mGenMar.mStar;
s1=L"aiml\\italian\\previsioni tempo\\"+s1+L".txt";
sQue=mGenMar.mAsgFil.readFileDir(s1);
Answer(sQue);
resetAction();
}

...

////////////////////////////////////// SET ARGOMENTO
if(mGenMar.mAction==L"updateargomento")
{
this->mArgomento.SetWindowTextW(mGenMar.mCurrDisc);
resetAction();
}

////////////////////////////////////// RICERCA INTERNET

if(mGenMar.mAction==L"cercagooglestar")
{
COleVariant noArg;
setGoogleSearch(mGenMar.mService);
resetAction();

}

////////////////////////////////////// LISTEN- DON'T LISTEN

if(mGenMar.mAction==L"startlisten")
{
this->mGenMar.flNoListen=false;
resetAction();
}

if(mGenMar.mAction==L"stoplisten")
{
this->mGenMar.flNoListen=true;
resetAction();
}

...

```


Tale caratteristica estesa ad ogni livello di gestione di una frase in ingresso rende il sistema aperto a miglioramenti e a un ulteriore sviluppo progressivo nel tempo.

Una versione ridotta di LUCY.1.0, a carattere tuttora sperimentale è presente online all'indirizzo: <http://www.sgarrincia.com/asbot/aswebchatbot.aspx>

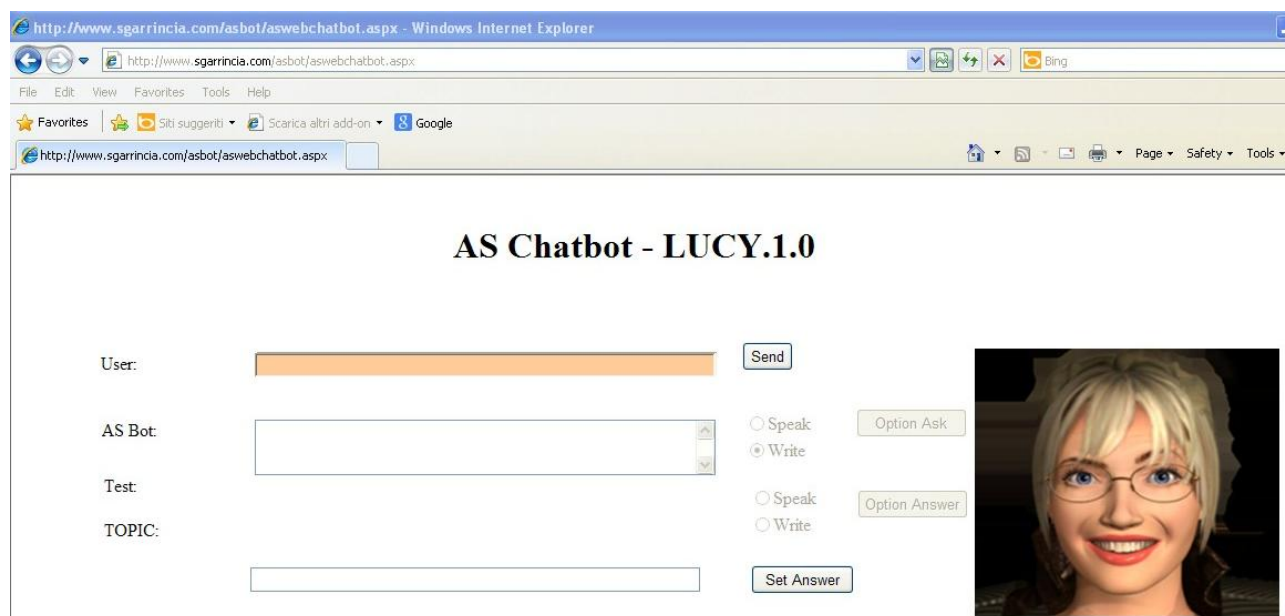


Fig. 73 - Lucy.1.0 online

Sintesi vocale

L'operazione di conversione di un testo elettronico in una sequenza acustica paragonabile alla voce umana (la fase finale della gestione di un chatbot, quella che fornisce l'eventuale realizzazione acustica di una risposta del sistema), non è così agevole come l'attuale disponibilità di sistemi TTS (Text To Speech) potrebbe far supporre.

La pronuncia di una parola scritta è difatti un'attività complessa, che richiede la capacità di programmare una sequenza fonica insieme al suo adeguato profilo prosodico e in relazione a condizioni esterne individuabili nel momento in cui l'atto viene effettuato. I vari computer generalmente a nostra disposizione presentano una serie di limitazioni fisiche, (come l'assenza di un adeguato apparato fono-articolatorio), che impediscono la necessaria resa spontanea e continua del parlato con voce naturale.

Inoltre, se anche si ipotizzasse di superare l'ostacolo dell'articolazione vera e propria con sequenze registrate, rimarrebbe la questione difficilmente gestibile di memorizzare la pronuncia di tutte le parole del vocabolario in ogni forma flessa, e soprattutto l'impossibilità di gestire la dimensione prosodica di una frase, con il risultato di letture comicamente innaturali.

È necessaria quindi la creazione di un modello formale in grado di superare tali difficoltà.

Diversi sono appunto i modelli di produzione del parlato studiati nel corso degli ultimi anni:

1. *Modello articolatorio*: attraverso questo modello si procede ad una simulazione dell'apparato fono articolatorio umano;

2. *Modello acustico*: si cerca di produrre un segnale che riproduca alcuni degli aspetti dell'onda sonora;

3. *Modello concatenativo*: si tenta di combinare porzioni di segnale prodotto dal parlare umano in maniera da produrre una emissione quanto più soddisfacente.

I primi modelli TTS erano detti *phoneme based*, ossia basati sull'inventario fonologico di ciascuna lingua: con un ridotto inventario di unità foniche si generava un segnale parlato che tuttavia non teneva conto dei fenomeni di coarticolazione, della transizione tra fonemi e della prosodia: il *parlato* ottenuto risultava del tutto innaturale, e a volte anche incomprensibile.

Oggi le unità di analisi su cui si basano i sistemi TTS sono più ampie e possono garantire una resa migliore.

Schematicamente il TTS è un'applicazione che riceve in ingresso un testo in forma elettronica, lo elabora e in output produce un'onda sonora parlata corrispondente al testo inserito. Le odierne applicazioni TTS presentano di solito un modello seriale, in cui vi sono vari livelli di trattamento dell'informazione linguistica successivamente sottoposti a diverse fasi di lavorazione. Uno schema indicativo è quello presente in figura 74.

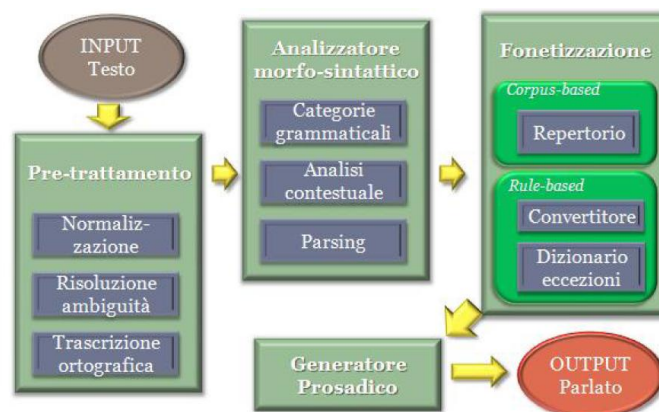


Fig. 74 - Schema di modello seriale TTS.

Il processo di generazione del parlare sintetizzato può essere difatti ridotto schematicamente a cinque fasi principali:

1) *Pre-trattamento*: questa fase serve a trasformare il testo in una forma utilizzabile dal calcolatore. La *normalizzazione* del testo converte in forma standard "leggibile" le diverse parti da sintetizzare, come numeri, sigle, acronimi, date e simboli (ad esempio trasforma "29" in "ventinove" e "€" in "euro"). A questa fase appartiene anche l'analisi della punteggiatura e la valutazione delle possibili ambiguità che influenzeranno il trattamento del testo, in particolare l'individuazione dei confini di una frase. Alla fine della fase, sarà presente un testo trascritto ortograficamente in una forma omogenea utilizzabile nella fase successiva.

2) *Analisi morfo-sintattica*: è effettuata da un modulo che provvede alla segmentazione del testo in unità morfografiche e/o sintattiche. Questa analisi serve per individuare la categoria grammaticale di ogni unità lessicale e a individuare la struttura e le relazioni tra i

sintagmi presenti nelle frasi del testo. Si tratta di una fase importante per trattare alcuni tipi di omografie, ma anche per fornire una base per le successive elaborazioni della struttura prosodica.

3) *Fonetizzazione*: dalla rappresentazione fonologica si realizza una rappresentazione fonetica che tiene presente degli allofoni e delle tendenze alla coarticolazione della lingua. Dato che ogni lingua manifesta notevoli differenze nel rapporto tra grafia e concrete fonie, esistono diversi sistemi di fonetizzazione, categorizzabili in due raggruppamenti principali:

a) *Rule -based*: operano su lingue in cui esiste un rapporto relativamente regolare tra fonemi e grafemi

b) *Corpora -based*: operano su lingue in cui la pronuncia di una parola non è strettamente connessa alla sua trascrizione grafica.

Per lingue come l'italiano, gran parte del lavoro per la fonetizzazione è realizzabile attraverso un convertitore grafema-fonema, che applica un numero finito di regole definite per trasformare i caratteri scritti in sequenze fonetiche. Le parole che non rispettano tali regole, come quelle di origine straniera, sono trattate in maniera separata e inserite in un dizionario fonetico delle eccezioni, che contiene direttamente la pronuncia delle parole che ne fanno parte. L'approccio, quindi, è ibrido, con l'uso di un sistema rule-based per trattare la maggior parte delle parole, e corpora-based, invece, per le eccezioni alle regole di pronuncia. Nel caso di lingue molto irregolari, come l'inglese, la conversione basata su regole gioca un ruolo marginale. Si preferisce, invece, un fonetizzatore basato su un dizionario di riferimento, più o meno esteso a seconda del modello, che contiene le diverse forme delle parole (parole testuali) in una trascrizione fonetica standard, simile dunque ai dizionari delle eccezioni. Per le lingue con una morfologia flessionale molto ricca a volte la rappresentazione è fatta in modo da separare inizialmente la radice dalla desinenza flessiva (velocizzando l'accesso al dizionario), per ricomporle in seguito alla sequenza fonetica.

4) *Generazione prosodica*: questa fase ha lo scopo di modulare l'intonazione del testo appena fonetizzato in modo che appaia naturale ed adeguata al tipo di enunciato da produrre. Si tratta di una fase molto complessa: si pensi a quante variazioni intonative aggiungono significato nella distinzione di frasi interrogative e affermative in italiano; o anche a quella sorta di accento sulle varie parole in rilievo all'interno di un enunciato.

5) *Sintesi del segnale*: consiste nella vera e propria generazione del segnale acustico, l'effettiva sintesi dell'onda sonora corrispondente al testo dato in input. Spesso per questa fase si usa uno *speech database*, ovvero una raccolta di segmenti del parlato già registrati, utili per produrre una sequenza continua.

In LUCY ho utilizzato il sistema di sintesi messo a disposizione dalla Microsoft e opportunamente configurato.

La seguente routine configura la *Speech API* di Windows:

```
bool CAsgSpeak::InitializeSapi() {
    if (FAILED(CoInitialize(NULL))) {
        AfxMessageBox(L"Error starting COM");
        return false;
    }
    HRESULT hRes = a_cpEngine.CoCreateInstance(CLSID_SpSharedRecognizer);

    if (FAILED(hRes)) {
        AfxMessageBox(L"Error starting SAPI");
        return false;
    }
}
```

```

    }
    hRes = a_cpEngine->CreateRecoContext(&a_cpRecoCtxt);

    if (FAILED(hRes)) {
        AfxMessageBox(L"Error creating context");
        return false;
    }
    hRes = a_cpRecoCtxt->SetNotifyWindowMessage(m_hWnd, WM_RECOEVENT, 0, 0);

    if (FAILED(hRes)) {
        AfxMessageBox(L"Error creating notification window");
        return false;
    }
    hRes = a_cpRecoCtxt->SetInterest(SPFEI(SPEI_RECOGNITION),
    SPFEI(SPEI_RECOGNITION));

    if (FAILED(hRes)) {
        AfxMessageBox(L"Error creating interest... ");
        return false;
    }
    hRes = a_cpRecoCtxt->CreateGrammar(0, &a_cpCmdGrammar);
    if (FAILED(hRes)) {
        AfxMessageBox(L"Error creating grammar");
        return false;
    }
    hRes = a_cpCmdGrammar->SetRuleState(NULL, NULL, SPRS_ACTIVE );
    if (FAILED(hRes)) {
        AfxMessageBox(L"Error setting rule state");
        return false;
    }
    }

    hRes = a_cpRecoCtxt->GetVoice(&a_cpVoice);

    if (FAILED(hRes)) {
        AfxMessageBox(L"Error creating voice");
        return false;
    }
    }

    hRes = a_cpVoice->SetInterest(SPFEI(SPEI_VISEME), SPFEI(SPEI_VISEME));

    if (FAILED(hRes)) {
        TRACE0("Error creating interest...seriously.\n");
        return FALSE;
    }
    }

    hRes = a_cpVoice->SetNotifyWindowMessage(m_hWnd, WM_VISEME_RECOEVENT, 0, 0);

    if (FAILED(hRes)) {
        TRACE0("Error setting notification window.\n");
        return FALSE;
    }
    }

return true;
}

```

Dopo le varie fasi di inizializzazione e di gestione della sincronia del microfono la sintesi è effettuata da un metodo:

```

void CAsgSpeak::Speak(CString sSentence)
{
    a_cpVoice->Speak(sSentence, SPF_DEFAULT, NULL);
}

```

Visemi

Il parlare umano è caratterizzato da elementi fondamentali che non sono definibili unicamente attraverso il sistema fonico-acustico-uditivo, e quindi mediante il solo segnale acustico. Difatti il parlare di ciascun individuo è sempre accompagnato da differenti espressioni del suo viso, per non dire dalla sua gestualità complessiva. Da questa constatazione, si sviluppa naturalmente l'idea di realizzare sistemi in grado di simulare un "parlato" multimodale, che non si avvalga solamente della modalità fonico-acustica. In questa direzione si muove la nascita di programmi in grado di fornire una sorta di sinestesia audio-visiva, e arricchire il contributo espressivo della comunicazione linguistica: parliamo delle cosiddette *teste parlanti*, rappresentazioni che congiungono alla normale sintesi vocale di un testo in forma scritta, l'animazione del disegno di un viso che simula i movimenti delle labbra, della mascella, della lingua e persino degli occhi durante il parlato.



Fig. 75 - Homepage del sito <http://www.sitepal.com/> che permette di creare degli avatar parlanti da inserire in una pagina web

Le teste parlanti sono realizzate con animazioni in tre dimensioni che possono produrre movimenti fluidi, grazie a una programmazione basata su dati reali acquisiti attraverso lettori ottici.

Ovviamente oggi giorno vengono anche sviluppate vere e proprie teste robotizzate in grado di realizzare le stesse prestazioni ma in vere e proprie strutture meccaniche.



Fig. 76 - Testa parlante.

Allo scopo di rendere anche la mia interfaccia graficamente più accettabile, ho utilizzato una gamma di immagini in formato “jpg” per riprodurre le varie espressioni facciali di un parlante nel momento in cui pronuncia una qualche frase. La tecnica utilizzata per riprodurre tali immagini fa ricorso ai cosiddetti *visemi* e alla maniera di riprodurre e animare un’espressione facciale in rapporto al linguaggio parlato, di cui abbiamo appena discusso.

Un *visema* risulta un’espressione *grafica* che corrisponde a un suono: ad esempio nella lettura delle labbra si sfrutta una particolare configurazione e posizione di quest’ultime per individuare il *fono* pronunciato da un parlante.

I visemi e i fonemi non sono in corrispondenza biunivoca. Spesso differenti fonemi corrispondono a un singolo visema, poiché tali fonemi risultano essere prodotti da una medesima configurazione delle labbra e degli organi fonatori esterni: ad esempio i fonemi /k/, /g/, /ŋ/, corrispondono al visema /k/, o anche /tʃ, ʃ, dʒ, ʒ/ sono relativi al visema /ch/, /t, d, n, l/ (viseme: /t/), e /p, b, m/ (viseme: /p/).

Parallelamente esistono molte differenze nella durata dei vari *speech act*, se si rapporta l’aspetto sonoro a quello visivo, e spesso, un determinato suono non può essere distinto se non con successive configurazioni o fotogrammi di animazione. La Speech Engine della Microsoft fornisce un insieme di funzioni per gestire la corrispondenza tra i fonemi inglesi e alcuni valori numerici utilizzabili per gestire il susseguirsi delle varie espressioni facciali. Per comodità nel mio chatbot non ho gestito correttamente tali corrispondenze (che per la lingua italiana avrebbe necessitato una traslazione di valori) ma ho soltanto sincronizzato i suoni emessi con dei movimenti casuali delle labbra (adottando una tecnica analoga a quella utilizzata nel doppiaggio dei film stranieri).



Fig. 77 - Esempio di visemi. All’immagine iniziale di sfondo si sovrappongono le altre immagini: l’ultimo visema individua la pronuncia della “f”, il visema precedente le consonanti “d, f, k, r, s”

Va per altro ribadito come quello dell’animazione facciale costituisce un altro settore di sviluppo fondamentale nell’ambito delle interfacce dialoganti, un settore in grado di conferire all’interazione uomo-macchina una più evoluta consistenza, dal momento che la comprensione di una frase è statisticamente più esatta allorquando sia l’aspetto acustico che visivo cooperano insieme.

I gradi di interazione possono variare a livello di durata delle espressioni:

```
<vhml>
  <person disposition="angry">
    First I speak with an angry voice and look very angry,
    <surprised intensity="50">
```

```

    but suddenly I change to look more surprised.
  </surprised>
</person>
</vhtml>

```

Va menzionato, a proposito, il Face Modeling Language (FML) che permette una più complessa gestione delle emozioni di una espressione facciale e della loro intensità:

```

<fml>
  <act>
    <par>
      <hdmv type="yaw" value="15" begin="0" end="2000" />
      <expr type="joy" value="-60" begin="0" end="2000" />
    </par>
    <excl event_name="kbd" event_value="" repeat="kbd;F3_up" >
      <hdmv type="yaw" value="40" begin="0" end="2000" event_value="F1_up" />
      <hdmv type="yaw" value="-40" begin="0" end="2000" event_value="F2_up" />
    </excl>
  </act>
</fml>

```

Fig. 78 - Esempio di *markup* presenti nel Face Modeling Language (FML)

La gestione bidimensionale dei visemi è, in ogni caso, analoga a quella tridimensionale (si sviluppa con gli stessi principi), utilizzabile nell'ambito della robotica, per configurare eventuali espressioni facciali (anche microespressioni) con quanto pronunciato dai vari robot. In questo caso esisterà un programma di gestione e interfaccia tra il messaggio testuale digitalizzato e gli elementi meccanici-fonico-articolatori del riproduttore finale.

11. WATSON e Jeopardy: lo stato dell'arte. Un approccio "concreto" alle nostre stesse problematiche.

14 febbraio 2011

Il sistema Watson sfida i campioni di *Jeopardy*, Ken Jennings e Brad Rutter, in una sfida di tre minuti. Jennings detiene il record per la più lunga serie vincente a questo gioco, mentre Rutter ha totalizzato il più alto montepremi della storia.



Fig. 79 - Watson alle prese con i due concorrenti Jennings e Rutter

16 febbraio 2011

...la seconda sfida tra Watson e i due campioni umani ha registrato un'altra vittoria per Watson. A questo punto la situazione è la seguente:

- Watson: 36.681 \$
- Jennings: 2.400 \$
- Rutter: 5.400 \$

...Watson è caduto in un paio di risposte relativamente semplici per un americano, come "Il suo più grande aeroporto è intitolato a un eroe della Seconda Guerra Mondiale". Il riferimento corretto è a Chicago, ma Watson ha menzionato Toronto.

17 febbraio 2011

...Watson vince definitivamente totalizzando 77.147\$ contro i 24.000\$ di Jennings e i 21.600\$ di Rutter.

Come abbiamo più volte ribadito le interfacce dialoganti hanno un ruolo primario nell'ambito della consultazione di un grande numero di documenti. Abbiamo nel corso di questa trattazione citato numerosi esempi di ricerca delle informazioni sottolineando come nell'ambito di tale ricerca, la velocità di individuare una risposta risulta fondamentale. Allo scopo di gestire la problematica della velocità congiunta alla correttezza l'IBM ha, di recente, intrapreso una sorta di avventura tecnologica cercando di realizzare un sistema in grado di rispondere correttamente alle domande del popolare programma televisivo Jeopardy e di gareggiare con un concorrente effettivo del programma. Ben nota risulta l'impresa del programma *Deep Blue* che nel 1996 fu in grado di battere il campione del mondo di scacchi, Garry Kasparov: le capacità del programma risultavano impostate su un insieme di regole relativamente semplici riducibili a modelli prettamente matematici, che il computer era in grado di gestire superbamente. Ma come abbiamo più volte rilevato le

regole in grado di modellare il linguaggio naturale umano sono molto più complesse. I sistemi attuali, generalmente, sono in grado di capire e rispondere a domande relativamente semplici espresse in un linguaggio altrettanto semplice (ad esempio “qual è la capitale d’Italia?”); un gioco a premi televisivo come *Jeopardy* sembrava inizialmente piuttosto al di sopra delle capacità dei sistemi più sviluppati. Inoltre, come abbiamo fatto rilevare, vincere a *Jeopardy* richiedeva l’individuazione di una risposta nel minor tempo possibile, nell’arco di pochi secondi. Prestazioni notevolmente al di fuori delle macchine più sviluppate.

Nell’immaginario collettivo, e come abbiamo più volte sottolineato, i computer dei film di fantascienza non sono altro che delle straordinarie interfacce dialoganti. Comprendono ciò che viene loro chiesto e forniscono il tipo di risposta, o la gamma di risposte che vengono richieste. Qualsiasi programma con tale scopo è costretto a far fronte alle consuete difficoltà che il linguaggio naturale presenta per un computer.

Innanzitutto le numerose estensioni nei vari significati delle parole. Quando una persona decodifica ciò che qualcun altro sta dicendo, possiamo facilmente individuare le varie allusioni, sfumature e connotazioni in ogni frase. Per un essere umano individuare ciò a cui si allude nella seguente frase: “il tipo di questo berretto è elementare, mio caro concorrente”; è estremamente facile. L’eco della parola “elementare”, suggerisce la famosa espressione “elementare, caro Watson” pronunciata da Sherlock Holmes e immediatamente nell’immaginario collettivo, la figura del detective inglese appare, così come essa è stata descritta nei vari romanzi e nei film ad essi ispirati (con la pipa e il celebre berretto di cervo in testa). Ma per un computer, non esiste una *semplice* maniera di identificare “è elementare, mio caro concorrente” come una allusione al personaggio. È possibile naturalmente un confronto tra diverse parole chiave e anche tra diversi sintagmi della frase, (in sostanza ciò che i motori di ricerca sono in grado di fare al giorno d’oggi), ma ciò non risulta abbastanza. Prima della comparsa degli articoli relativi a questa “impresa” era possibile sperimentare quanto detto su Google. Inserendo la stessa frase, era individuabile il riferimento all’espressione di Sherlock Holmes ma nessuna allusione al berretto.

In ogni caso, anche se un computer riuscisse ad individuare che la domanda riferita all’allusione appena espressa risulta “quale tipo di cappello indossava Sherlock Holmes?”, le informazioni presenti nella base dati del sistema potrebbero non essere memorizzate in una maniera tale che abiliti il computer a estrarre una precisa risposta. Per anni, gli scienziati informatici hanno costruito sistemi di risposta a domande creando database specializzati, nei quali i cosiddetti “fatti” riguardanti la conoscenza del mondo erano registrati e collegati insieme. Si potrebbe agire allo stesso modo con la figura di Sherlock Holmes, costruendo un database che include connessioni tra frasi o parole chiave e il suo cappello oppure il violino che di solito suona nel tempo libero. Ma quel database risulterebbe piuttosto ristretto e riferito a un contesto ben specifico. Estendendo il nostro interesse saremo ben presto dinanzi all’impossibile, utopistico compito di organizzare tutte le informazioni conosciute dall’uomo: sarebbe un’impresa titanica. Tale problematica è conosciuta come il cosiddetto *collo di bottiglia*:¹⁴⁹ anche se si dovesse superarlo ci troveremmo a confrontarci con ulteriori problemi: Se il nostro database contiene solo fatti che inseriamo manualmente, risulta non funzionante ogni volta che noi proponiamo una domanda a qualcosa che non è presente in quel materiale (come ho già spiegato nella descrizione dei chatbot). Non esiste alcun sistema per annotare manualmente un database in

¹⁴⁹ In ingegneria, il collo di bottiglia o col d’oca è un fenomeno che si verifica quando le prestazioni di un sistema o le sue capacità sono fortemente vincolate da un singolo componente.

grado di includere le risposte a ogni possibile allusione, simile a quelle presenti in una trasmissione come un gioco a premi.

Un grande cambiamento nell'ambito dell'intelligenza artificiale si è verificato per altro, sostanzialmente negli ultimi dieci anni, quando gli scienziati informatici hanno iniziato ad utilizzare, come abbiamo visto, considerazioni statistiche per analizzare una enorme quantità di documenti, comprendenti libri e notizie di cronaca, esempio.

Si è cominciato a scrivere algoritmi che in grado di gestire ogni stringa di parole in modo statistico cercando di calcolare su queste basi *quanto* i sintagmi (nel caso della domanda riguardante Sherlock Holmes) “cappello di cervo”, “Professor Moriarty”, “oppio” oppure “violino” siano correlati tra di loro, e *quanto* siano distanti da altre sequenze di parole come ad esempio “Campionato di calcio”.

In questo modo si potrebbe presentare a un computer una domanda che non menzioni Sherlock Holmes direttamente, ma permetta di concludere che Sherlock Holmes sia il probabile soggetto delle nostre allusioni, analogamente ad altre centinaia di concetti.

In teoria questa sorta di approccio in passato non era praticabile a livello strettamente fisico. I computer non erano abbastanza veloci, le memorie non abbastanza grandi e in ogni caso non era affatto facile inserire milioni di documenti all'interno di un computer (si pensi ai primi computer per i quali l'input era effettuato attraverso schede perforate).

Di recente la situazione è notevolmente cambiata. I computer sono divenuti drasticamente più economici, e il numero di testi on-line è esploso nel momento in cui milioni di persone hanno iniziato a scrivere blog e voci enciclopediche come quelle presenti nel sito di “Wikipedia”. Il numero di testi trasformati in formato digitale è aumentato quindi in quantità vertiginosa: gli strumenti precedentemente ideati dagli studiosi dell’NLP (in grado di gestire soprattutto l'aspetto morfologico e sintattico del linguaggio) sono divenuti sempre più importanti.

Ciò nonostante anche i migliori tra questi sistemi qualora dovesse partecipare a una trasmissione come Jeopardy otterrebbero dei risultati deludenti. Gli esseri umani riescono ad adattarsi a gran parte delle domande e in una questione di secondi, acquisiscono l'informazione la compongono, la interpretano, pervengono alle giuste conclusioni all'interno della propria memoria e quindi forniscono l'informazione ricercata, pressoché in maniera istantanea.

Effettuiamo un esempio concreto su una delle possibili tecniche adottate da Jeopardy e consideriamo la traccia “Fu perdonato dal Presidente l'8 settembre 1974”

In una prima analisi il sistema individua immediatamente il riferimento a Nixon. Per valutare poi se “Nixon” è la risposta migliore, Watson esegue una serie di procedimenti: inserisce la risposta all'interno della frase originale “Nixon fu perdonato dal presidente l'8 settembre 1974” e quindi ripercorre il suo database per controllare l'esistenza di ulteriori riscontri in grado di supportare l'ipotesi. L'individuazione di una frase come “Ford perdonò Nixon l'8 settembre 1974” confrontata con la frase di partenza contribuisce a porre l'ipotesi “Nixon” come la più probabile delle risposte.

Altri tipi di elaborazioni e algoritmi possono essere utilizzati per altri tipi di domande. Ad esempio si possono effettuare dei confronti specifici per quanto riguarda i riferimenti spazio temporali, e dai successivi riscontri proporre l'eventuale risposta. Se dinanzi alla traccia “Nel 1594 intraprese l'attività di esattore delle tasse in Andalusia” le risposte generate sono “Thoreau” e “Cervantes” un successivo confronto con l'anno di nascita di

Thoreau, (1817) esclude quest'ultimo dalle scelte e Cervantes diviene la risposta maggiormente quotata.

Tra gli sviluppi possibili di Watson potrebbe risultare una sua applicazione in ambito medico. Si potrebbe immaginare un ospedale che fornisce la base di conoscenza di un sistema analogo a Watson con ogni forma di documento a carattere medico esistente, o appena pubblicato, e quindi ottenere risposte a domande durante i momenti di emergenza. In ambito medico come sappiamo la gamma delle nuove procedure, delle nuove medicine, delle nuove conoscenze si sviluppa così velocemente che i medici non sono in grado di aggiornarsi alla velocità con cui le novità prendono il proprio corso. Come abbiamo visto l'ambito medico è uno dei principali ambienti in cui i sistemi dialoganti possono essere utilizzati. Ma un sistema come Watson potrebbe anche essere sfruttato (vedremo meglio nel capitolo conclusivo le concrete possibili applicazioni future delle interfacce dialoganti) per allestire dei virtuali call centers dove un computer parlerebbe direttamente ai clienti e risulterebbe il primo a interagire con eventuali reclami o emergenze.

Va ribadito, per altro, che il sistema è in grado di rispondere a domande riguardanti fatti accertati ed è completamente impotente dinanzi a domande che investono la sfera delle ipotesi e delle teorie, come ad esempio “qual è la migliore compagnia quotata in borsa esistente sulla quale investire del denaro?” o “quanto durerà il processo di pace nel medio oriente?”. Il sistema è cioè in grado di comparare le varie richieste con la base di conoscenza a sua disposizione, scegliere eventuali risposte attraverso criteri statistici, comporre una stringa di testo in grado di fornire una risposta.

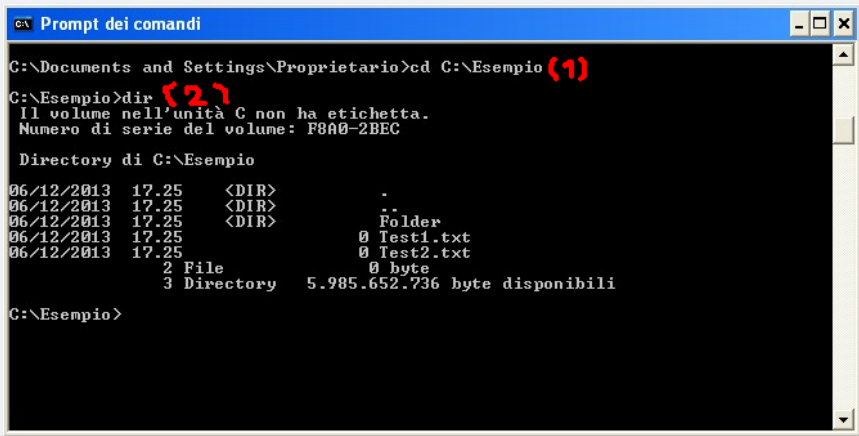
Ciò nonostante, secondo gli scienziati che hanno lavorato al progetto, Watson è in grado di simulare, quantunque in modo artefatto, alcune delle maniere con le quali il nostro cervello elabora il linguaggio. Difatti la moderna neuroscienza ha scoperto che il nostro cervello “lavora” essenzialmente in parallelo: usa cioè differenti sue parti simultaneamente, miliardi di neuroni vengono attivati indipendentemente ogni qualvolta parliamo o ascoltiamo delle parole. Il procedere di Watson attraverso una molteplicità di ipotesi, generate durante una stessa richiesta lascerebbe supporre un comportamento analogo a quello che i bambini effettuano ogni qual volta cercano di individuare il significato delle parole, o ragionare sulle strutture grammaticali di una frase.

12. Prospettive

Nel corso di questo studio abbiamo più volte ribadito le numerose difficoltà incontrate nella realizzazione di una vera e propria Intelligenza Artificiale o di una interfaccia dialogante in grado di elaborare il *nostro* linguaggio naturale.

A causa di tali problematiche si è riscontrato negli ultimi anni, una sorta di cambiamento di prospettiva: si è cercato di restringere il campo di interesse dei sistemi da costruire, per realizzare applicazioni tali, da essere impiegate con profitto nelle attività degli esseri umani (i vari software di riconoscimento vocale e di traduzione automatica sono un esempio concreto di questo discorso).

Parimenti, come abbiamo più volte osservato si sta verificando un nuovo approccio per sfruttare pienamente le potenzialità di un elaboratore. Ricordiamo che i primissimi computer erano programmabili attraverso schede perforate o cilindri rotanti, e solo in un secondo tempo si sono prodotti dei dispositivi con una interfaccia testuale: ma anche in questo caso, i comandi che bisognava dare al computer costringevano l'utente ad imparare una nuova lingua, quella del sistema operativo installato nel computer, e programmare le varie istruzioni in base alla morfologia, alla sintassi e alla semantica del sistema.



```

C:\Documents and Settings\Proprietario>cd C:\Esempio (1)
C:\Esempio>dir (2)
Il volume nell'unità C non ha etichetta.
Numero di serie del volume: F8A0-2BEC

Directory di C:\Esempio

06/12/2013 17.25 <DIR>      .
06/12/2013 17.25 <DIR>      ..
06/12/2013 17.25 <DIR>      Folder
06/12/2013 17.25             0 Test1.txt
06/12/2013 17.25             0 Test2.txt
                2 File             0 byte
                3 Directory 5.985.652.736 byte disponibili

C:\Esempio>
```

Fig. 80 - Esempio di alcune istruzioni del sistema operativo DOS

Nella figura 80 l'istruzione "cd C:\Esempio" (1) apre la directory "Esempio" come in ambiente Windows si può aprire tale directory, cliccando sull'icona corrispondente. L'istruzione "dir" (2) elenca invece i file ("Test1.txt" e "Test2.txt") e le cartelle ("Folder") presenti nella directory.

Nel corso degli anni, con l'intenzione di "antropomorfizzare" l'utilizzo e la gestione del calcolatore da parte degli utenti finali, appaiono le prime interfacce grafiche (dette anche GUI, Graphic User Interface) che introducono una serie di nuove metafore visive nell'ambito del funzionamento del computer. L'utente non deve più trovare il comando nella lingua del sistema operativo, ma può operare direttamente sui cosiddetti menu e sulle ormai ben note icone.

I sistemi operativi prodotti prima dalla Apple e poi dalla Microsoft, attualmente in uso, sono basati su tali forme di interfacce: gli elementi fondamentali sono le icone, e le finestre.

L'introduzione delle interfacce grafiche è difatti concomitante alla diffusione dei computer nella nostra realtà contemporanea; ciò nonostante, anche questo approccio non risulta privo di problemi e inconvenienti.

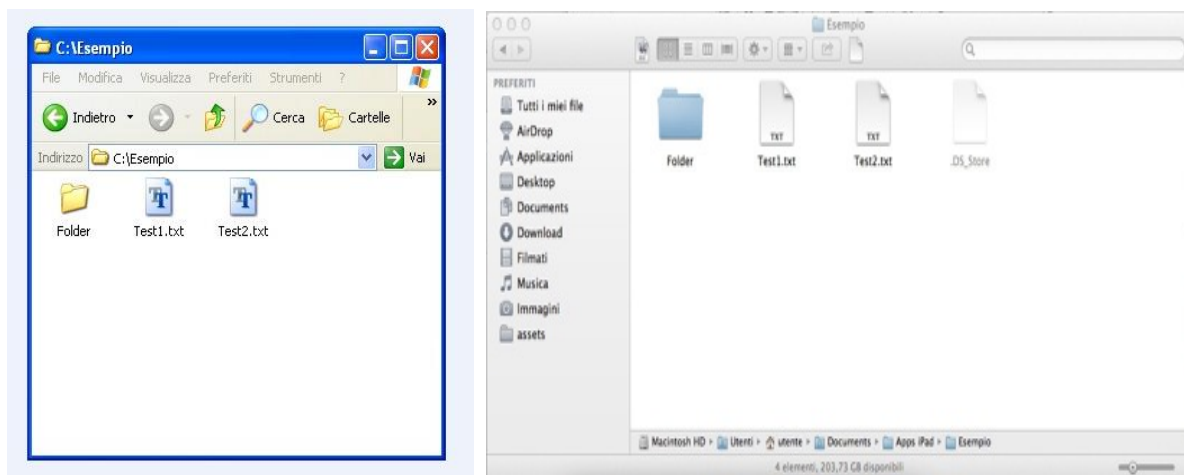


Fig. 81 - Icone e cartelle del sistema Windows XP e OS X 10.8

Le interfacce grafiche, soprattutto in sistemi informativi articolati, tendono a crescere di complessità e a richiedere lunghi periodi di apprendimento. Anche i più semplici programmi presentano spesso delle difficoltà durante le varie fasi di ricerca e le stesse icone non sempre sono in grado di significare in modo esplicito le caratteristiche del file a cui sono associate.



Fig. 82 - Esempio di un Menu del programma Word della suite Office 2007

Per superare le difficoltà insite nella gestione di sistemi informativi complessi, stiamo perciò assistendo, paradossalmente, a un ritorno verso lo sfruttamento delle funzionalità del comando testuale: lo strumento più usato su internet è il motore di ricerca Google che risulta, di fatto, un comando testuale. A partire dalla casella di testo possiamo inserire i nostri comandi che verranno interpretati dal motore di ricerca in base ai propri algoritmi.



Fig. 83 - Casella di testo di Google

Sviluppando questo approccio, congiuntamente all'utilizzo delle interfacce grafiche, in futuro si potrà fruire di un un assistente sempre a nostra disposizione in grado di rivoluzionare la nostra interazione con il mondo dell'informatica: chi si impegnerebbe in una ricerca tra le pagine web, di eventuali servizi o prodotti quando può semplicemente esporre le proprie necessità a un ipotetico assistente e aspettare di essere esaudito? Chi, in

un negozio, consulterebbe indici e cataloghi per recuperare da solo il prodotto da acquistare quando potrebbe chiedere l'aiuto di un commesso? Le interfacce dialoganti non richiedono all'utente alcuno sforzo e sono perfette per gli utenti non esperti.

I vari motori di ricerca si stanno perciò orientando decisamente verso lo sfruttamento di comandi testuali: per esempio in Google si può scrivere “define: chatbot” per ottenere una lista di definizioni o in Yahoo si può scrivere “tempo atmosferico a Roma” per avere notizie sul tempo nella capitale. In altre parole i motori di ricerca si stanno gradualmente trasformando in motori di risposta (answering machine) che utilizzano proprio i comandi testuali.

Lo stesso accade nei sistemi per la gestione di grandi quantità di file come Gmail, uno dei sistemi di posta più diffusi: le cartelle non sono più presenti o utilizzabili ma il ritrovamento delle mail avviene solamente attraverso comandi di testo.

Attualmente le interfacce testuali sono anche in linea con le ricerche più avanzate che le grandi compagnie di software stanno portando avanti: si cerca cioè di realizzare interfacce in grado di consentire la gestione di un dispositivo, senza la tastiera, e impartendo tramite la voce i comandi nel nostro linguaggio naturale: raggiungendo così le potenzialità dei computer della fantascienza come in *Star Trek* o *2001: Odissea nello Spazio*.

L'antropomorfizzazione delle interfacce conosce cioè questo ulteriore sviluppo: dopo il passaggio dalle schede perforate alle interfacce testuali (con le tastiere e il mouse), si ha una ulteriore evoluzione con le interfacce grafiche, e quindi una successiva fase nella quale si potrà ridurre, se non eliminare, l'utilizzo dei mouse e delle tastiere come è attualmente concepito.

La tecnologia dei chatbot si sviluppa secondo, difatti, tale orientamento.

Ricordiamo a proposito, nuovamente la figura di Alan Turing che col suo articolo *Computing Machinery and Intelligence* nel 1950 individuava (come abbiamo ricordato) proprio nel dialogo tra uomo e macchina un sistema per misurare il grado di intelligenza (in questo caso quella di un computer), secondo parametri antropomorfi.

Uno degli aspetti interessanti del test di Turing consiste proprio nel fatto che la macchina non deve essere veramente *intelligente*, ma deve apparire tale.

L'approccio di Turing ha stimolato, in ogni caso, fin dai primi anni dell'era informatica, lo studio dei chatbot.

Ricordiamo i vari esperimenti su cui abbiamo discusso: ELIZA di Weizenbaum, SHRDLU, Jabberwacky o anche programmi come Alice e anche Watson.

Tutti questi esempi sono, genericamente, sistemi di azione-reazione: non c'è una reale comprensione del linguaggio ma un insieme di risposte a diversi stimoli precodificati.

Difatti, per anni i ricercatori impegnati nella costruzione di chatbot hanno realizzato principalmente programmi in grado di parlare e scrivere come un essere umano: obiettivo sicuramente legato al desiderio di superare il Test di Turing.

Il Loebner Prize ha proprio questa impostazione: il premio, istituito da un mecenate attivo come produttore di spettacoli a Broadway, mette tuttora in palio un milione di dollari per il sistema dialogante in grado di superare il test, e, nella certezza che nessun computer è vicino a tale possibilità, un premio minore, ma dell'ordine di grandezza di poche migliaia di dollari, per il migliore programma.

Il Loebner Prize ha rivelato, nonostante queste premesse poco lusinghiere, tipiche di una lotteria per imbonitori, esiti piuttosto curiosi e meritevoli, a questo punto, di una discussione: nonostante nessun computer in fin dei conti, e alla lunga, si è avvicinato minimamente alla possibilità di “battere” un essere umano in fatto di “umanità” si sono

verificati dei singoli casi, delle singole sessioni in cui alcuni particolari giudici hanno ritenuto un computer più umano del vero essere umano. Il paradosso che ne scaturisce presenta una dublice interpretazione: si potrebbe supporre che il giudice stesso abbia fallito il test di Turing, o anche che l'essere umano esaminato non sia riuscito a convincere il giudice di essere abbastanza umano.

Uno degli accadimenti più celebri nelle leggende del premio, si è verificato con un professore di letteratura inglese interrogato su Shakespeare: il giudice alla fine lo aveva classificato come non umano poiché secondo la sua ipotesi nessun essere umano avrebbe potuto conoscere tante informazioni sul celebre drammaturgo, come quelle possedute dal professore. Il professore venne dichiarato non umano per eccesso di conoscenza: difatti il premio è stato più volte vinto dal programma che riusciva a manipolare meglio la comunicazione e il dialogo (come nel caso di MegaHAL). Paradossalmente i programmi che cercavano di evadere le domande ostentando disinteresse, (con risposte tipo “non mi interessa, perché non parliamo di cinema?” oppure “perché dovremmo sprecare il nostro tempo parlando di queste cose, qual è la tua squadra preferita?”) ottenevano i giudizi migliori: mantenendo questo atteggiamento, sebbene per un tempo ristretto, il computer riesce a sembrare sorprendentemente umano.

Un approccio con queste caratteristiche ha comunque sicuramente fatto sviluppare metodologie e nuovi linguaggi, ma ha anche finito per concentrare la ricerca dei chatbot sulla pura conversazione: una sorta di dilettantistico dialogo da *party*, non specifico e non finalizzato.

L'approccio delle interfacce dialoganti più moderne utilizza invece le tecnologie e i metodi dei chatbot tradizionali ma li concentra su domini più ristretti. L'efficacia di un chatbot, o meglio di un assistente virtuale, la sua concreta utilità, si sviluppa in corrispondenza del dominio di competenze del *bot* stesso.

Possiamo classificare i vari domini secondo due caratteristiche: i domini aperti e i domini chiusi.

I chatbot che hanno come obiettivo la semplice discussione, il passatempo tipico di una *chatroom*, devono avere un dominio di conoscenze aperto: essere cioè capaci di capire *possibilmente* ogni cosa, di rispondere in modo sensato a ogni tipo di domanda. Molti dei chatbot sviluppati con questo scopo raggiungono in parte questo obiettivo ma nel loro obiettivo onnicomprensivo, finiscono per evitare la concretezza di un'argomentazione e non dire niente di rilevante. Si è parlato di “risposta del politico” per descrivere queste caratteristiche. Vengono sfruttati alcuni appigli forniti dalla conversazione per esporre un proprio discorso ed evitare domande e argomenti più specifici e concreti.

Il Test di Turing non è ancora superabile perché richiede un complesso sistema linguistico e semantico, una conoscenza dei significati e dei significanti, dei modi di dire, della conoscenza umana a ogni livello e della logica.

Per questo motivo la ricerca sui chatbot si è, in qualche modo, arenata proprio come, negli anni 80-90, le ricerche sulla cosiddetta “Strong AI” che cercava cioè di creare programmi veramente intelligenti hanno dovuto compiere dei passi indietro. Attualmente, il nuovo approccio, quello della “Weak AI”, cerca invece di creare programmi che abbiano “semplicemente” delle funzionalità intelligenti.

L'approccio ridotto opera quindi su una porzione del linguaggio e in ambiti estremamente chiusi e limitati, legati ad esempio alle tecnologie VoiceXML¹⁵⁰ e a tutti i call center e agli information center automatici.

Alcuni di questi sistemi sono piuttosto efficaci ma utilizzabili solamente per compiere attività estremamente specifiche: transizioni bancarie, prenotazioni di biglietti ecc.

Una strada intermedia che alcuni ricercatori e alcuni prodotti commerciali stanno cercando di esplorare si sviluppa nel tentativo di sfruttare le conclusioni di entrambi gli orientamenti: non si cerca di costruire un sistema per dialogare su uno specifico compito (sistema task-oriented) né si tenta di costruire un chatbot capace di parlare su *tutto* (test di Turing) ma si creano programmi in grado di condurre una conversazione libera su un dominio specifico, su un argomento specifico: si costruisce cioè un assistente virtuale su quell'argomento.¹⁵¹

Ma quali sono a questo punto le reali prospettive di utilizzo?

Tra gli esempi più immediati possiamo menzionare alcuni programmi già presenti in molte aziende e soprattutto nella pubblica amministrazione: in un sito online è possibile simulare la presenza di un impiegato esperto e sempre disponibile che offre indirizzi, orari, riferimenti e tutto ciò che può servire per chi è alla ricerca di informazioni specifiche.

Tale sistema potrebbe essere impiegato nelle ore di ufficio per smaltire automaticamente pratiche semplici (si pensi all'attuale diffusione delle macchine automatiche per l'emissione dei biglietti ferroviari o metropolitani: nell'ambito dell'informazione, tali sistemi potrebbero costituire un analogo approccio rivoluzionario.)

Un altro ambito di utilizzo potrebbe verificarsi nei servizi a carattere sanitario: le varie aziende potrebbero offrire nuovi e importanti servizi informativi: potrebbero difatti offrire al pubblico chatbot specializzati in tematiche specifiche come per esempio informazioni su varie malattie (diabete, malattie genetiche, cardiopatie, celiachia ecc.) o su tematiche legate alla sanità e alla salute in generale (sessualità, alcolismo, droghe, disabilità, alimentazione ecc.).

Un chatbot, un assistente virtuale in grado di rispondere a domande specifiche potrebbe sostituire per certi versi la professionalità di un medico generico e favorire gli utenti meno solleciti a una visita o a un consulto da parte di un altro essere umano.

Ulteriori esempi sono legati alla amministrazione militare: esercito e forze armate, o in altri ambiti a carattere civile: erario, pratiche legate al lavoro, immigrazione ecc.: un chatbot può intervistare, presentare contenuti e dare indicazioni puntuali (come presentare una richiesta, dove, in quale ufficio) all'utente interessato.

Un'interfaccia dialogante può fornire un validissimo aiuto in tutti i negozi virtuali e i siti di e-commerce. Un chatbot può assumere le veci di un commesso virtuale e aiutare (dando informazioni o aprendo opportune pagine web) un eventuale utente.

Un chatbot può essere molto utile anche per le aziende che non vendono direttamente online: l'assistente virtuale, può offrire assistenza ai clienti sul servizio o il prodotto per aiutarne l'utilizzo, risolvere problemi o raccogliere consigli o critiche. In area marketing,

¹⁵⁰ VoiceXML (VXML) acronimo di *Voice eXtensible Markup Language*, definito dal World Wide Web Consortium (W3C), rappresenta lo standard in formato XML per la creazione di dialoghi interattivi tra una persona ed un computer.

La piattaforma VoiceXML lavora in modo analogo a un browser HTML: i documenti VoiceXML vengono scaricati da un server web, interpretati e trasformati in voce da un Voice Gateway residente sul computer dell'utente finale.

¹⁵¹ Kerly A., Hall Phil, Bull Susan. *Bringing chatbots into education: Towards natural language negotiation of Open Learner Models*. In Ellis R., Allen T, Tuson A., *Applications and Innovations in Intelligent Systems XIV* (2007)

potrebbe dialogare su tematiche legate all'azienda presentando il prodotto, aumentandone la cosiddetta *brand awareness*, e facendo crescere il legame, anche emotivo, tra un cliente e il marchio della ditta.

Ogni sito di *ecommerce* ha la sua organizzazione dei prodotti, la sua nomenclatura specifica e la sua interfaccia grafica: per quanto il sito sia funzionale non sarà mai possibile che tutti gli utenti trovino subito tutto ciò che serve loro. Spesso l'esplorazione, soprattutto per gli utenti non esperti, diventa lunga, frustrante e inefficace, e si conclude con una mancata vendita per il negozio. Con un chatbot che agisce come un assistente virtuale, il potenziale cliente può evitare di capire tassonomie (l'oggetto ricercato è in questo o in quel sottomenu?) e icone, e chiedere direttamente, scrivendo, quello che gli serve.

A volte i clienti sanno di voler acquistare un "panno in microfibra" ma molto più spesso sanno che hanno bisogno di "un qualcosa che aiuti a togliere la polvere dagli scaffali".

A volte si può aver bisogno di un "pianale modulare da parete" ma molto più spesso si necessita "di qualcosa per riporre i libri e le riviste". In altre parole, i siti di *ecommerce*, generalmente, presentano i prodotti ma non intercettano i bisogni del cliente come, per esempio, un buon commesso reale sa fare: un commesso virtuale, attraverso il linguaggio naturale, può comprendere le necessità generiche del cliente e indirizzarlo al prodotto o ai prodotti di cui ha bisogno.

Un utente con un bisogno, in un sito normale, è costretto a tradurre le proprie esigenze in forme adatte alle categorie e al linguaggio utilizzato nel sito. Per esempio la domanda "come pago la bolletta tramite la banca?" deve essere tradotta, in un sito normale, in "domiciliazione delle utenze": con un chatbot l'utente può utilizzare un linguaggio più semplice che verrà tradotto dagli algoritmi della eventuale interfaccia.

Infine un'interfaccia dialogante potrebbe fornire informazioni sull'azienda, tranquillizzare sulla sicurezza dell'acquisto elettronico, descrivere le opzioni di pagamento e dare tutte le informazioni legate alla compravendita online.

Le interfacce dialoganti possono facilmente coesistere con la normale interfaccia di un sito e quindi non interferire con gli eventuali clienti abituali ed esperti che già conoscono il sito stesso o preferiscono visitarlo nel modo classico.

Un esempio presente sul web di questa tipologia di chatbot, come già detto, è quello dell'IKEA. Il chatbot dell'IKEA è un esempio di commesso virtuale e mette in atto, parzialmente, quanto descritto nei precedenti paragrafi.

Tutte le applicazioni appena descritte creano una maggiore soddisfazione del cliente e dell'utente (customer - user satisfaction) e si traducono immediatamente in una riduzione dei costi in quanto gli operatori umani sono alleggeriti di una gran parte del loro lavoro.



Fig. 84 - Versione svedese dell'assistente virtuale di IKEA

Intrattenimento e non solo

Una forma, la più semplice, per comunicare informazioni su un prodotto, su un servizio o su un marchio è quello di intrattenere i clienti e i potenziali clienti su tematiche legate al prodotto: in questo senso un chatbot può dialogare divertendo e istruendo sulle tematiche scelte dall'area marketing. Un assistente virtuale, per la sua innovativa modalità di comunicazione uno a uno, può costruire un rapporto diretto con l'utente (anche attraverso canali non comuni per il marketing).

Con le migliaia di messaggi pubblicitari che ci giungono quotidianamente è sempre più difficile convogliare con efficacia un messaggio riguardante un prodotto: gli spot, di qualsiasi natura essi siano, rischiano infatti di essere dimenticati o, addirittura, di non essere percepiti.

Ciò nonostante, una *storia* "ben costruita" è in grado di catturare e mantenere l'attenzione dell'ascoltatore: le narrazioni, con il loro aspetto emotivo, sono in grado di creare un legame profondo con l'ascoltatore e questo legame emozionale facilita la memorizzazione e l'interiorizzazione del messaggio. Tali prerogative possono essere sfruttate a livello di marketing ma non solo: oltre ai video, ai vari prodotti multimediali gli assistenti virtuali rappresentano un modo nuovo, e altamente innovativo, di raccontare storie come nessun altro mezzo, se non un essere umano, può fare: appunto il dialogo.

La ricerca nel campo dell'intelligenza artificiale applicata alla narrazione viene spesso denominata *narrative intelligence* o *interactive storytelling*¹⁵² e studia come un computer può raccontare una storia e interagire, nello stesso tempo, con un utente. Questa area di ricerca (legata attualmente, come si può intuire, al mondo dell'intrattenimento e dei video game) è molto vitale e dinamica ma ha prodotto, finora soltanto elaborazioni per lo più teoriche.

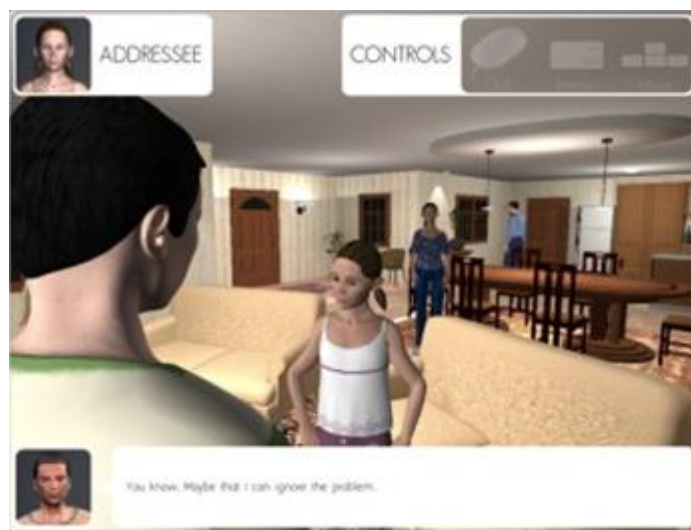


Fig. 85 - Animazione in una storia interattiva.

Un'interfaccia dialogante, opportunamente progettata, può essere ritenuta una forma di *interactive storytelling* in quanto è in grado di raccontare una storia e reagire

¹⁵² Si consulti l'indirizzo <http://iris.interactive-storytelling.de/> un esempio di website per la creazione di storie in modo interattivo

contemporaneamente alle richieste dell'ascoltatore: in questo modo è possibile associare la forza emotiva (didattica, mnemonica, ecc.) della narrazione con il coinvolgimento interattivo.

Ci sono poi esempi di marketing legato allo storytelling (per esempio Coca-Cola e BMW) anche se nessuno utilizza in modo specifico l'approccio dell'interactive storytelling.

Naturalmente le interfacce dialoganti possono essere utilizzate per unire l'intrattenimento alla formazione e all'educazione. Per il pubblico più giovane (molto legato alle chat e all'instant messaging) è possibile creare personaggi e cosiddetti avatar¹⁵³ che traggono la loro personalità e le loro tematiche dai protagonisti dei programmi più celebri (ad esempio anche dai cartoni animati o dai fumetti).

L'utilizzo di moderni sistemi in grado di svolgere funzioni di *tutoring* a vari livelli e soprattutto a livello scolastico sono del resto stati studiati e messi alla prova in un ipotetico confronto con l'attività di un eventuale docente.

Negli ultimi anni i sistemi di Intelligent Tutoring hanno avuto un notevole riscontro e investono i più disparati campi della didattica. Questi sistemi sono in genere costruiti su un modulo di apprendimento in grado di classificare le capacità del discente e proporle a quest'ultimo attraverso un grado di oggettività che può indurre quest'ultimo a una migliore comprensione del suo reale livello di preparazione e facilitarne il successivo progresso.

Da questi presupposti è facile ipotizzare la realizzazione di sistemi costruiti interamente sulla capacità di interagire con gli studenti attraverso il linguaggio naturale: si parla di *Negotiated Learner Modelling*¹⁵⁴ nel quale l'utente viene fatto conversare all'interno di un ben definito tessuto di argomenti in grado di sfruttare informazioni estraibili da appositi database e ovviamente anche dalla rete.

Siti personali

In modo molto simile a un servizio informativo, un professionista può creare un'interfaccia dialogante riferita al proprio dominio di conoscenza: in questo modo potrà segnalare e proporre un sito (o un indirizzo di instant messaging) che diventa una sorta di linea diretta con i propri clienti e utenti. Le professioni interessate possono essere numerosissime: medici, avvocati, consulenti, giardinieri, cuochi, insegnanti, esperti di ogni tipo e genere.¹⁵⁵

L'assistente virtuale potrà essere messo a disposizione di tutti o accessibile solamente a pagamento.

¹⁵³ L'avatar è una immagine scelta per rappresentare la propria utenza in comunità virtuali, luoghi di aggregazione, discussione, o di gioco on-line. La parola, derivata dal sanscrito, appartiene alla tradizione induista, nella quale significa incarnazione, assunzione di un corpo fisico da parte di un dio (Avatar: "Colui che discende"): per traslazione metaforica, nel gergo di internet si intende che una persona reale sceglie di mostrarsi agli altri attraverso una propria rappresentazione, una sorta di incarnazione: un *avatar* appunto.

¹⁵⁴ Si consideri l'articolo "The Potential for Chatbots in Negotiated learner Modelling: A Wizard-of-Oz Study" di Alice Kerly e Susan Bull all'indirizzo internet:

<http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.131.6034&rep=rep1&type=pdf>; Kerly A., Bull S., The Potential for Chatbots in Negotiated Learner Modelling, in Ikeda M. Ashley K., Chan T-W, *Intelligent Tutoring Systems*, 8th International Conference, pp. 443-452, Springer-Verlag, Berlino (2006).

¹⁵⁵ Addirittura, nonostante l'aspetto surreale dell'eventualità, oracoli e maghi per la lettura di tarocchi, oroscopi ecc.,

In altri contesti, uomini politici o i vari partiti partiti potrebbero utilizzare le interfacce dialoganti per comunicare con i propri sostenitori e potenziali elettori.

Tale approccio può essere utilizzato per creare dei chatbot che siano degli alterego di un politico specifico: un chatbot può infatti fornire all'elettore la sensazione di dialogare direttamente con il proprio candidato (o possibile candidato), rivolgergli le domande sulle tematiche che più sente importanti senza dover leggere ciò che non gli interessa, chiedergli della sua vita personale o dei suoi interessi privati: attraverso un chatbot è possibile creare un rapporto diretto, paradossalmente emotivo, con i vari potenziali elettori.

La necessità per i cosiddetti "personaggi famosi" (cantanti, gruppi musicali, calciatori, attori e attrici, veline, personaggi televisivi, artisti, ecc.) di essere presenti sul web è considerata poi un ulteriore campo di applicazione. In quest'ottica la creazione di un chatbot in grado di dialogare direttamente con degli ammiratori è una possibilità unica e fortemente innovativa di costruire e mantenere viva l'attenzione e l'interesse sul proprio personaggio.

L'ascolto

In tutte le applicazioni appena descritte le interfacce dialoganti sono state illustrate per la loro possibilità di dialogare, di rispondere, in modo personale ed efficace. Ma c'è un altro aspetto estremamente importante comune a tutti gli assistenti virtuali: la possibilità di registrare quello che gli utenti hanno da dire. In altre parole un'interfaccia dialogante offre un nuovo modo di ascoltare i vari singoli utenti, capire che cosa vogliono sapere o che cosa, per loro, è importante e bello.

È per altro anche possibile creare dei chatbot che non simulano una persona reale ma un personaggio immaginario, storico o letterario: gli obiettivi possono essere come al solito, ovviamente molteplici come per esempio intrattenere, insegnare, utilizzare il personaggio per fare marketing, parodiare, promuovere un film o un libro.

Un esempio interessante di personaggio virtuale è quello del cantante John Lennon.¹⁵⁶

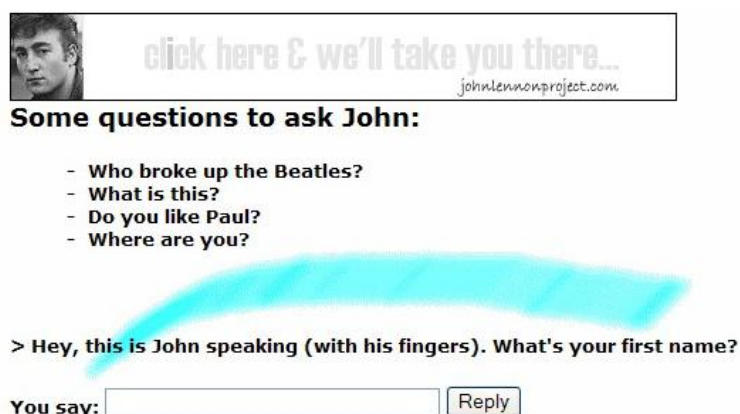


Fig. 86 - Chatbot presente sul sito del John Lennon Project

¹⁵⁶ Il John Lennon Artificial Intelligence Project è presente al sito: <http://triumphpc.com/johnlennon/>

Colloqui e interviste di lavoro

Un'interfaccia dialogante può essere utilizzata per condurre, via web, delle interviste o dei colloqui: in questa maniera è possibile raccogliere, in modo molto simile ad un colloquio di lavoro tradizionale, molte informazioni puntuali e centrate sulle necessità dell'azienda stessa.

L'interfaccia dialogante può infatti crearsi, con domande e “ascolti”, un modello cognitivo dell'utente e consigliarlo sui suoi gap di competenze, formazione da seguire, crescita professionale da intraprendere ecc. Questa tipologia di assistente virtuale è stata già realizzata (e testata con successo) per una Università statunitense (Kerly A., Phil Hall b, Susan Bull. (2007). *Bringing chatbots into education: Towards natural language negotiation of open learner models*).

Durante il dialogo con l'utente, l'interfaccia può attivare immagini o schemi che illustrano parti del prodotto o i modi corretti di operare. Queste attività si rivelano particolarmente utili con prodotti ricchi di informazione e che richiedono, per essere sfruttati al massimo grado, una certa competenza da parte dell'utente. Dialogare con assistenti virtuali evita all'utente di dover leggere difficili manuali e offre anche all'azienda dei grossi vantaggi: l'azienda può infatti evitare o ridurre grandemente il servizio di assistenza postvendita e raccogliere le domande più frequenti dei clienti stessi.

Simulazioni e narrazioni formative.

Un chatbot può simulare infine, a scopi formativi, il comportamento di diverse tipologie di persone. In ambito medico può interpretare il ruolo di una persona convalescente per esercitare i futuri medici a raccogliere le necessarie informazioni terapeutiche; in un'azienda generica potrebbe essere sfruttato per simulare un ipotetico cliente (a ruoli rovesciati rispetto alle precedenti applicazioni) per formare i venditori a conversare efficacemente con i propri interlocutori, oppure per formare il personale dei vari call center.

Risultati e futuri sviluppi della ricerca

L'obiettivo principale di questa dissertazione è stato la presentazione di una nuova realtà che al momento attuale sta rivoluzionando il rapporto uomo macchina con un impatto forse analogo a quello in cui i personal computer e internet hanno rivoluzionato il mondo del lavoro e dei rapporti interpersonali negli ultimi decenni.

Ho presentato alcuni aspetti fondamentali del Natural Language Processing allo scopo di costruire una interfaccia dialogante evidenziando i vari approcci effettuati nel corso degli ultimi decenni per realizzare tale meta e proponendo alcune soluzioni nella versione di chatbot da me sviluppata *Lucy.1.0*

Mi sono concentrato in una versione in lingua italiana e oltre al programma Windows ho reso disponibile una versione ridotta del chatbot online, in lingua italiana, collegata ad alcune sue versioni ulteriori (sempre online) aggiornabili e sviluppabili dai vari utenti che vorranno avere un proprio personalissimo assistente virtuale in rete.

Ho studiato e approfondito alcune delle specifiche tecniche per l'elaborazione del linguaggio naturale realizzando e aggiornando alcuni algoritmi di tokenizing, tagging, e clustering.

Ho costruito diverse librerie di software in grado di elaborare un testo digitale o un corpus di testi digitali per le più svariate esigenze: dalla semplice ricerca di neologismi, alla definizione della metrica in una successione di versi, alla costruzione di eventuali ontologie.

A questi programmi si aggiungono altri algoritmi di Information Extraction applicati a testi digitali e a pagine web (indicizzabili in maniera automatica) allo scopo di costruire una base di conoscenza per la mia interfaccia dialogante o per altri scopi.

Nel corso di questo lavoro ho constatato, per altro, le difficoltà che i principali ricercatori attualmente interessati nell'ambito dell'NLP affrontano da decenni e quanto sia complessa la creazione di algoritmi utili per gestire il linguaggio umano. Problematiche varie che convergono nella attuale incapacità di realizzare sistemi in grado di possedere una consistente parvenza di intelligenza artificiale nel contesto linguistico.

Ho per altro evidenziato come, in contesti ben specifici, le nuove tecnologie, congiunte alle tecniche di machine learning e di apprendimento non supervisionato possono garantire una funzionalità anche pratica per i sistemi in questione.

Siamo in grado cioè di realizzare sistemi di riconoscimento e di sintesi vocale piuttosto affidabili e possiamo far riferimento a una *smisurata* mole di dati digitali accessibili con velocità attualmente paragonabili a quelle con cui il nostro cervello recupera le informazioni al suo interno.¹⁵⁷

L'esempio del successo del sistema della IBM, Watson, è una dimostrazione del significato di queste asserzioni.

L'obiettivo finale, del resto, in base ai criteri da me esposti è proprio quello di organizzare tali dati in modo simile a quello con cui un essere umano li organizza ed elabora, creare cioè quella sorta di dispositivo in senso chomskyiano in grado di comprendere e generare le frasi di una determinata lingua.

La realizzazione del mio chatbot Lucy ha avuto ed ha sostanzialmente questo obiettivo, analogo a quello di numerosi centri di ricerca in tutto il mondo che operano nella prospettiva descritta all'interno di questo lavoro.

Lucy, una volta definita la base di conoscenza su cui far riferimento, e le strategie per gestire uno specifico settore di interesse, è già adesso in grado di operare come assistente virtuale nei più disparati campi (dalla prenotazione di un biglietto aereo, al servizio di consultazione in una ricerca storica o letteraria, al semplice conversare per diletto).

È in grado di memorizzare i vari dialoghi, costruire e apprendere modelli linguistici in base alle conversazioni avute, semplificare la gestione del sistema operativo in cui è inserita.

Ciò nonostante la sua capacità di adattarsi a nuove situazioni non supervisionate è sostanzialmente limitata ed è appunto questo uno degli obiettivi futuri nel perfezionare tale chatbot. Il nuovo lavoro di aggiornamento, verrà perciò effettuato congiuntamente alla realizzazione di versioni sempre più complesse di Lucy in grado di essere utilizzate online (gestendo le immense risorse di internet come una sorta di motore di ricerca) e anche offline nell'ambito di sistemi robotizzati.

I vari criteri di valutazione numerica del chatbot, sono stati trascurati in questa ricerca poiché il mio scopo è al momento quello di definire un possibile percorso da sviluppare

¹⁵⁷ Mentre i neuroni del nostro cervello operano con periodi dell'ordine di millisecondi, i vari processori operano generalmente con periodi di nanosecondi e diventano progressivamente più veloci. La "differenza" è già in un rapporto di 10^6 . le macchine intelligenti potranno presto pensare milioni di volte più veloci del cervello umano. Analogo discorso ovviamente, può essere fatto per la capacità nell'estensione della memoria: relativamente limitata nel cervello umano, pressoché illimitata in un cervello elettronico.

progressivamente: di definire cioè metodi che potranno essere applicati in questa seconda fase nella quale sarà aggiornata e ampliata la base di conoscenza effettiva, facendo ricorso a ulteriori annotazioni manuali e a un sempre maggiore sfruttamento dei vari corpora, dell'italiano in circolazione.

Parallelamente dovrò sviluppare e integrare nuove e più articolate tecniche di *machine learning* a livello semantico, sintattico e pragmatico in grado di migliorare e rendere il più possibile "autosufficiente" il chatbot.

Questo genere di studi, nel periodo in cui stiamo vivendo ha una portata davvero rivoluzionaria.

Negli anni sessanta abbiamo "assistito" all'ottimismo entusiastico dei linguisti e degli informatici e alle loro successive delusioni. Abbiamo osservato la nascita, il declino, e la successiva rinascita, dell'Intelligenza Artificiale intesa come disciplina; negli anni settanta, si è maturato il miraggio onnicomprensivo delle reti neurali, e successivamente è cresciuta l'attenzione degli scienziati verso una disciplina come la neuroscienza cognitiva. Abbiamo vissuto poi le due grandi rivoluzioni nell'ambito dell'informatica (ricordiamo: quella del personal computer e quella di internet) e concretamente osservato la particolare tempistica con cui tali cambiamenti sono avvenuti: proprio in questi anni, inoltre, si stanno producendo le prime automobili di serie provviste di controlli vocali.

Nel 1996 un computer ha battuto il campione del mondo di scacchi; nel 2011 un computer è riuscito a battere due validi concorrenti in un gioco a premi di cultura generale.

Si può realisticamente affermare che ci troviamo nelle vicinanze di una nuova stravolgente frontiera del progresso e dell'evoluzione del genere umano.

Bibliografia

- Allen James, *Natural Language Understanding*. Benjamin/Cummings, Redwood City (1995).
- Allen James, e Perrault Ray, “Analyzing intention in utterances”. *Artificial Intelligence* (1980).
- Allen James, Miller Bradford, Ringger Eric, Sikorski Tiresa, “A robust system for natural spoken dialogue”. *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics*, pp. 62-70 (1996).
- Allen, James, Schubert Lenhart K., Ferguson George et al., The trains project: a case study in building a conversational planning agent. *Journal of Experimental and Theoretical AI*, 7: pp. 7-48 (1995).
- Alpaydin E., *Introduction to Machine Learning*, MIT Press, Cambridge (2004).
- Appelt Douglas, Hobbs Jerry, Bear John, Israel David, Tyson Mabry, “FASTUS: a finite-state processor for information extraction from real-world text”. *Proceedings of IJCAI-93* 1172–8 (1993).
- Appelt Douglas, *Planning English Referring Expressions*, Cambridge University Press, New York (1985).
- Austin J. L., *How to Do Things with Words*, Harvard University Press, Cambridge (1962).
- Baayen H. *Word Frequency Distributions*,. Kluwer, Dordrecht (2001).
- Baayen H., *Analyzing Linguistic Data: A Practical Introduction to Statistics*. Cambridge University Press, Cambridge (2008).
- Banko Michele, e Brill Eric, “Scaling to very very large corpora for natural language disambiguation”. *Proc. ACL*. 337, 519, 520 (2001).
- Baroni M., Bernardini S., “BootCaT: Bootstrapping corpora and terms from the web”, *Proceedings of Fourth International Conference on Language Resources and Evaluation*, pp. 1313-1316 (2004).
- Bellman R. E., *Dynamic Programming*, Princeton University Press, Princeton (1957).
- Belz A., “Automatic generation of weather forecast texts using comprehensive probabilistic generation-space models. *Natural Language Engineering*” 14 (4) (2007).
- Berners-Lee Tim, Hendler J., Lassila O., “The semantic web”, *Scientific American* 284 (5), pp. 24-30 (2001).

- Biber D., *Variation Across Speech and Writing*, Cambridge University Press, Cambridge (1988).
- Bikel D., M., Schwartz R., Weischedel R. M.. “An algorithm that learns what’s in a name”, *Machine Learning*, 34 (1), pp. 211-231 (1999).
- Bloomfield L., *Language*, Henry Holt, New York (1933).
- Bratko I., *PROLOG Programming for Artificial Intelligence*. Addison–Wesley, Reading (2011).
- Brill E., “Transformation-based error-driven learning and natural language processing: A case study in part-of-speech tagging”, *Computational Linguistics*, 21 (4), pp. 543-565 (1995).
- Brown P., Pietra S. D., Pietra V. D., Mercer R., “The mathematics of statistical machine translation: Parameter estimation”, *Computational Linguistics*, 19 (2), pp. 263-311 (1993).
- Carletta J., “The reliability of a dialogue structure coding scheme”, *Computational Linguistics*, 23 (1), pp. 13-32 (1997)
- Chomsky N., *Syntactic Structures*, Mouton, L’Aia/Parigi (1957).
- Chomsky N., *Aspects of the Theory of Syntax.*, MIT Press, Cambridge (1965).
- Chomsky N., *Lectures on Government and Binding*. Foris, Dordrecht (1981).
- Clark H., Schaefer E. F., “Contributing to discourse”. *Cognitive Science*, 13, pp. 259-294 (1989).
- Clarke Arthur C., *The Sentinel: Masterworks of Science Fiction and Fantasy*. Berkley Books (1983)
- Clippinger John, *Speaking with Many Tongues: some Problems in Modeling Speakers of Actual Discourse*, Teleos, Cambridge (1977).
- Colby K., *Comments on human-computer conversation*, in Wilks, Y., *Machine Conversations*. pp. 5-8 (1999).
- Crichton Michael, *Sphere*, Knopf (1987)
- Davey Anthony, *Discourse Production*, Edinburgh University Press, Edinburgh (1979).

- Dominus M. J., *Higher-order Perl: a guide to program transformation*. Elsevier, Amsterdam (2005).
- Earley J.. “An efficient context-free parsing algorithm”. *Communications of the ACM* 13 (2), pp. 94-102 (1970).
- Elkan C., “The paradoxical success of fuzzy logic”, in *Proceedings of the Eleventh National Conference on Artificial Intelligence (AAAI-93)*, pp. 698-703. MIT Press, Cambridge, 1993.
- Gazdar G., *Pragmatics*, Academic Press, New York (1979).
- Gazdar G., Mellish C., *Natural Language Processing in Lisp : an Introduction to Computational Linguistics*, Addison-Wesley, Wokingham (1989).
- Gazdar G., Mellish C., *Natural Language Processing in Prolog : an Introduction to Computational Linguistics*. Addison-Wesley, Wokingham (1989).
- Ge, N., Hale J., Charniak E., “A statistical approach to anaphora resolution” in *Proceedings of the Sixth Workshop on Very Large Corpora*, pp. 161-170, Montreal (1998).
- Gibbon D., Moore R., Winski R., *Handbook of Standards and Resources for Spoken Language Systems*, Mouton de Gruyter, Berlino (1997).
- Gödel Kurt, “Über Formal Unentscheidbare Sätze der *Principia Mathematica* und Verwandter System, I”, *Monatshefte für Mathematik und Physik*, 38, pp. 173-198 (1931).
- Goldman Neil M., *Conceptual Generation*, in Schank Roger C., Riesbeck Christopher K., *Conceptual Information Processing*. American Elsevier, New York (1975).
- Grice P. H., “Logic and Conversation”, *Speech Acts: Syntax and Semantics Volume 3*, pp. 41-58. Academic Press, New York (1975).
- Grice P. H., *Studies in the Way of Words*, Harvard University Press, Cambridge (1989).
- Grice Paul, *Logic and conversation in Syntax and semantics 3: Speech acts*, a cura di P. Cole, Academic Press, New York (1975).
- Grimes Joseph, *The Thread of Discourse*. Mouton (1975).
- Grosz B., Sidner C., “Attention, Intentions and the Structure of Discourse”, *Computational Linguistics*, 12(3), pp. 175-204.
- Guha R. V., Lenat D., “Cyc: A Midterm Report”, *AI Magazine*, pp. 33-59 (1990).
- Halliday Michael, *Spoken and Written Language*, Deakin University Press, Victoria (1985).

- Halliday M., *Introduction to Functional Grammar*, Edward Arnold, Londra (1994).
- Halliday M., Hasan R., *Cohesion in English*, Longman, Londra (1976).
- Hammond, M., *Programming for linguists : Java™ technology for language researchers*, Blackwell, Oxford (2002).
- Handke J., *The Structure of the Lexicon*, Mouton de Gruyter, Berlino (1995).
- Hearst Marti, “TextTiling: Segmenting Text into Multi-Paragraph Subtopic Passages”, *Computational Linguistics*, 23 (1), pp. 33-64 (1997).
- Hofstadter Douglas R., *Gödel, Escher, Bach: un’Eterna Ghirlanda Brillante*, Basic Books (1979)
- Hutchins W. J., *Early years in machine translation: Memoirs and biographies of pioneers*, John Benjamins, Amsterdam (2000).
- Kerly A., Hall Phil, Bull Susan. *Bringing chatbots into education: Towards natural language negotiation of Open Learner Models*. In Ellis R., Allen T, Tuson A., *Applications and Innovations in Intelligent Systems XIV* (2007)
- Kerly A., Bull S., The Potential for Chatbots in Negotiated Learner Modelling, in Ikeda M. Ashley K., Chan T-W, *Intelligent Tutoring Systems*, 8th International Conference, pp. 443-452, Springer, Berlino (2006).
- Lewis D. D., Catlett J., “Heterogeneous uncertainty sampling for supervised learning”, in *Proceedings of the 11th International Conference on Machine Learning*, New Brunswick (1994).
- Hopcroft J., Ullman J., *Introduction to Automata Theory, Languages, and Computation*, Addison-Wesley, Reading (1979).
- Huang X., Acero A., Hon H.-W., *Spoken Language Processing: A Guide to Theory, Algorithm and System Development.*, Prentice-Hall, Englewood Cliffs (2001).
- Jacques Robin, McKeown Kathleen R., “Empirically designing and evaluating a new revision-based model for summary generation”, *Artificial Intelligence* 85, pp. 135-79.
- Jelinek F., *Continuous speech recognition by statistical methods*, Proceedings of the IEEE 64 (4), 532-556 (1976).
- Jurafsky D., Martin J. H., *Speech and Language Processing*, Prentice Hall (2009).

- Jurafsky D. et al., “Automatic detection of discourse structure for speech recognition and understanding”, in *Proceedings of the 1997 IEEE Workshop on Speech recognition and Understanding*, p. 88-95, Santa Barbara (1997).
- Levinson Steve, *Pragmatics*, Cambridge University Press, Cambridge (1983).
- Kay M., “The mind system”, in *Natural Language Processing*, pp. 155-88. Algorithmics Press, New York (1973).
- Kozima Hideki, “Text segmentation based on similarity between words”, *Proceedings of the 31st Annual Meeting (Student Session) of the Association for Computational Linguistics* (1993)
- Knuth D. E., *The Art of Computer Programming.*, Addison-Wesley, Reading (1969).
- Lappin S., Leass H. J., “An algorithm for pronominal anaphora resolution”, *Computational Linguistics* 20(4), pp. 535-561 (1994).
- Levinson, S., *Pragmatics*, Cambridge University Press, Cambridge (1983).
- Levinson, S., *Presumptive Meanings*, MIT Press, Cambridge (2000).
- Mann William, Thompson Sandra, “Rhetorical structure theory: a theory of text organization.” RS-87-190, *Information Sciences Institute* (1987)
- Manning, C., Schutze H., *Foundations of Statistical Natural Language Processing*, MIT Press, Cambridge (2003).
- Manning C. D., Carpenter B., “Probabilistic parsing using left corner language models”, in *5th International Workshop on Parsing Technologies*, Boston (1997).
- Manning, C. D., Raghavan P., Schutze H., *Introduction to Information Retrieval*, Cambridge University Press, Cambridge (2008).
- Marcu, D., “The rhetorical parsing of unrestricted texts: A surface-based approach”, *Computational Linguistics* 26 (3), pp. 395-448 (2000).
- McCarthy J., Lehnert W., “Using decision trees for coreference resolution”, in *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, Denver (1995).
- McDonald D. D., *Natural Language Generation*. In Dale R., Moisl H., Somers H., *A handbook of natural language processing: techniques and applications for the processing of language as text*, pp. 147-179, Marcel Dekker, New York (2000).

- McKeown K., "Discourse strategies for generating natural-language text", *Artificial Intelligence* 27, pp. 1-41 (1985).
- McKeown K., *Text Generation: Using Discourse Strategies and Focus Constraints to Generate Natural Language Text*, Cambridge University Press, Cambridge (1985).
- Meteer Marie, *Portable Natural Language Generation using SPOKESMAN*, Rensselaer Polytechnic Institute, New York (1992).
- Minsky M., *A framework for representing knowledge*, in Winston P. H., *The Psychology of Computer Vision*, pp. 211-277, McGraw Hill, New York (1975).
- Moser L., "Simulating Turing machines in DATR", *Cognitive Science Research Paper*, CSRP 241, Brighton (1992).
- Nilsson N., *Artificial intelligence - A New Synthesis*, Morgan Kaufmann, San Mateo (1998).
- Ovidio, *Le metamorfosi*, X, 243-297
- Paterson John, *The Hamlyn Pocket Dictionary of Wines*. Hamlyn (1980)
- Pereira F., Warren D., „Definite clause grammars for natural language analysis”, *Artificial Intelligence* 23, 231-278 (1980).
- Power R., "The organization of purposeful dialogs", *Linguistics*, 17, pp. 105-152 (1979).
- Russell S., Norvig P., *Artificial Intelligence: A Modern Approach*. Prentice Hall, New York (2010).
- Sacks H., Schegloff E. A., Jefferson G., "A simplest systematics for the organization of turn-taking for conversation", *Language*, 50(4), pp. 696-735
- Schank R. C., *Conceptual Information Processing*, Amsterdam (1975).
- Schegloff E. A., "Sequencing in conversational openings", *American Anthropologist*, 70, pp. 1075-1095 (1968).
- Searle J. R., *A taxonomy of illocutionary acts*, in Gunderson K, *Language, Mind and Knowledge, Minnesota Studies in the Philosophy of Sciences*, Vol. VII, pp. 344-269, University of Minnesota Press, Amsterdam (1975) .
- Sells, P. (1985). Lectures on Contemporary Syntactic Theories. CSLI Lecture Notes 3. Stanford University Press.
- Shannon, C. E. (1948, July, October). A mathematical theory of communication. The Bell System Technical Journal 27, 379-423, 623-656.

- Shelley Mary, *Frankenstein, o il moderno Prometeo* (1818).
- Stalnacker R. C., *Assertion*, in Cole P., *Pragmatics: Syntax and Semantics, Volume 9*, pp. 315-332, Academic Press, New York (1978).
- Turing A. M., “Computing machinery and intelligence”, *Mind*, 59:433–460 (1950).
- Watson Mark, *Practical Artificial Intelligence Programming with Java* (2001).
- Weizenbaum Joseph, *Computer Power and Human Reason: From Judgement to Calculation*, W.H. Freeman and company, San Francisco (1976).
- Winograd T., Flores F., *Understanding computers and cognition: a new foundation for design*, Ablex, Norwood (1986).
- Winograd T., *Understanding Natural Language*, Academic Press (1972).
- Wittgenstein L., *Tractatus logico-philosophicus*, Routledge (1922).
- Woods W. A., “Transition network grammars for natural language analysis”, *Communications of the ACM* 13(10), pp. 591-606 (1970).
- Zhao S., Grishman R., “Extracting relations with integrated information using kernel methods”, *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL-05)*, pp. 419–426, Ann Arbor (2005).
- Zipf G. K., *The Psycho-Biology of Language*. Houghton Mifflin, Boston (1935).

Sitografia

Chatbot:

<http://www.alicebot.org/>

<http://megahal.alioth.debian.org/>

<http://www.eloisa.it/>

<http://dpkeller.awardspace.com/ChatBot/chatBot.htm>

http://www.pandorabots.com/pandora/talk?botid=f5d922d97e345aa1&skin=custom_input

http://www.nlg-wiki.org/systems/Table_of_NLG_system_builders

<http://www.doriana82.com/>

<http://www.cs.cmu.edu/afs/cs/project/ai-repository/ai/areas/classics/parry/>

[Hci.stanford.edu/winograd/shrdlu.](Hci.stanford.edu/winograd/shrdlu)

http://en.wikibooks.org/wiki/Computer_Information_Systems_in_Education/Chapter_3/Section_1_--_Student_Response_Systems

<http://www.jabberwacky.com/>

<http://www.artificial-solutions.com/about-artificial-solutions/>

NLG:

<http://inf.abdn.ac.uk/research/stop/>

http://books.google.it/books?id=Ex6xZlxvUywC&pg=PA122&hl=it&source=gbs_toc_r&cad=4

<http://code.google.com/p/simplenlg/>

<http://www.abdn.ac.uk/ncs/departments/computing-science/skillsum-314.php>

<http://www.columbia.edu/cu/record/23/20a/summary.html>

<http://libots.sourceforge.net/>

Conoscenza comune:

<http://www.cyc.com/>

<http://commons.media.mit.edu/en/>

DAMSL:

<http://www.cs.rochester.edu/research/speech/damsl/RevisedManual/>

IE:

<http://www.ai.sri.com/natural-language/projects/fastus.html>

Stemming:

<http://snowball.tartarus.org/algorithms/italian/stemmer.html>

NLP Tutorials:

<http://clic.cimec.unitn.it/marco/>

Intelligenza Artificiale:

www.hs-weingarten.de/~ertel/aibook.

<http://aima.cs.berkeley.edu>

<https://leanpub.com/javaai>

Vari:

http://nltk.googlecode.com/svn/trunk/nltk_data/index.xml

<http://iris.interactive-storytelling.de/>

<http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.131.6034&rep=rep1&type=pdf>

<http://triumphpc.com/johnlennon/>

<http://www.w3.org/XML/>

<http://www.cogsci.princeton.edu/~wn/>.

Filmografia

2001: Odissea nello spazio (1968), regia di Stanley Kubrick.

Sfera (1998), regia di Barry Levinson

Corto circuito (1986), regia di John Badham

Blade Runner (1982), regia di Ridley Scott.

Wargames (Giochi di guerra) (1983), regia di John Badham

Indice

<i>Introduzione: paradigmi di interfacce dialoganti</i>	p. 4
1. Riferimenti storici	p. 12
2. Il riconoscimento vocale	p. 21
3. Preprocessing Tools	p. 25
4. Information Extraction	p. 43
5. Teoria del discorso e applicazioni	p. 68
6. Natural Language Generation	p. 76
7. La conoscenza comune	p. 96
8. Interfacce dialoganti	p. 103
9. Esempi di Chatbot	p. 139
10. Lucy.1.0	p. 159
11. WATSON e Jeopardy: lo stato dell'arte. Un approccio "concreto" alle nostre stesse problematiche.	p. 177
12. Prospettive	p. 181
<i>Risultati e futuri sviluppi della ricerca</i>	p. 191
Bibliografia	p. 194
Sitografia	p. 201
Filmografia	p. 202