INFORMATICA

Intelligenza artificiale e soft computing

Applicazioni pratiche per aziende e professionisti

Lorenzo Schiavina
Giancarlo Butti

FRANCOANGELI

Informazioni per il lettore

Questo file PDF è una versione gratuita di sole 20 pagine ed è leggibile con

La versione completa dell'e-book (a pagamento) è leggibile con Adobe Digital Editions. Per tutte le informazioni sulle condizioni dei nostri e-book (con quali dispositivi leggerli e quali funzioni sono consentite) consulta cliccando qui le nostre F.A.Q.

Am - La prima collana di management in Italia

Testi advanced, approfonditi e originali, sulle esperienze più innovative
in tutte le aree della consulenza manageriale,
organizzativa, strategica, di marketing, di comunicazione,
per la pubblica amministrazione, il non profit...

Lorenzo Schiavina
Giancarlo Butti

Intelligenza artificiale e soft computing

Applicazioni pratiche per aziende e professionisti

FRANCOANGELI

Progetto grafico di copertina di Elena Pellegrini

Questo libro è dedicato
a Camilla,
Anita e
Costanza
che essendo millennial saranno fuzzy native
Lorenzo

A mia moglie
per i 25 anni trascorsi insieme
e al nostro Book, che non ci vede,
ma che ci sente con il cuore.
Giancarlo

Indice

Parte prima

Parte prima

1 Intelligenza artificiale e soft computing

1. Un po' di storia dell'informatica (per capire come siamo arrivati al soft computing)

Quando nel 1968 Marshal McLuhan parlava del *villaggio globale* ben pochi avrebbero creduto che la sua visione si sarebbe concretizzata nel giro di una cinquantina di anni, anche se l'era dell'elaboratore era ormai concretamente iniziata il 16 febbraio 1946 quando Presper Eckert e John Mauchly avevano presentato ufficialmente ENIAC (Electronic Numerical Integrator And Computer), il primo elaboratore elettronico sviluppato per le esigenze dell'esercito americano.

Il '900 fu infatti l'inizio dell'era degli elaboratori elettronici, la cui realizzazione fu propiziata dallo sviluppo della matematica nell'ultima parte dell'800 e nei primi anni del '900.

La prima vera nascita dell'informatica è da collocare nella seconda metà dell'800, quando il matematico e filosofo inglese Charles Babbage, progettò la prima macchina analitica, che a tutti gli effetti è da considerarsi la prima concettualizzazione di un elaboratore.

A titolo di curiosità rileviamo che con Charles Babbage collaborò anche lady Ada Lovelace, figlia del poeta Gordon Byron; sembra che sua madre, terrorizzata dalla possibilità che la figlia seguisse le orme del padre e diventasse poeta, la spingesse verso la matematica.

Circa un secolo dopo, l'esercito americano sviluppò in suo onore il linguaggio di programmazione Ada.

Un altro personaggio dell'800 che avrebbe contribuito grandemente allo sviluppo dell'elaboratore fu George Boole[1], un logico e matematico, anch'egli inglese.

1. Prima di lui, Gottfried Wilhelm (von) Leibniz aveva già utilizzato la numerazione binaria.

George Boole sviluppò l'algebra (o logica) booleana che, grazie anche al contributo di Claude Shannon, diventerà successivamente la base logica per la realizzazione operativa dell'elaboratore.

Non proprio associabili all'informatica come tale, devono sicuramente essere ricordati gli studi di logica che alla fine dell'800 prepararono gli sviluppi matematici che furono la base per la realizzazione nella seconda metà del '900.

Occorre quindi ricordare i contributi alla logica matematica di Friedrich Ludwig Gottlob Frege, considerato il padre del pensiero formale, su cui si basarono successivamente lo sviluppo dei linguaggi di programmazione.

Infine, tra i pre-informatici dell'800 non si può dimenticare Herman Holerith, un ingegnere di origine tedesca che, in occasione del censimento americano del 1880, sviluppò una macchina tabulatrice in grado di elaborare la massa di informazioni raccolte; per questo lavoro, Holerith utilizzò per la prima volta le schede perforate, che sarebbero diventate il supporto principale per la memorizzazione dei dati dei primi elaboratori.

All'inizio del'900, anche se non direttamente associabile agli elaboratori, occorre ricordare David Hilbert, che presentò al congresso internazionale di Parigi i 23 più importanti problemi matematici da approfondire; il lavoro dei matematici che si occuparono di questi problemi è risultato, in parecchi casi, un tassello importante della realizzazione fisica o concettuale di componenti del mondo dell'informatica.

Il Novecento fu (dal punto di vista dell'elaboratore) un secolo brevissimo, a causa degli eventi bellici che lo caratterizzarono.

Tuttavia, la seconda guerra mondiale fu un momento di grande spinta, dato che lo sforzo bellico portò ad un importante contributo alle tecniche matematiche soprattutto da parte dell'Inghilterra che per prima sviluppò un nuovo ramo applicativo della matematica (operational reasearch o operating reasearch; in italiano ricerca operativa) per contrastare l'attività degli u-boat tedeschi.

L'insieme di tutto questo scenario formò la base per arrivare allo sviluppo di ENIAC.

Due altri personaggi devono essere citati per completare il quadro dello sviluppo delle tecnologie che hanno portato alla realizzazione del villaggio globale: John Von Neumann (che fu assistente di David Hilbert) e Alan Turing (che aveva lavorato nel gruppo inglese di scienziati coinvolti nella seconda guerra mondiale) e fu fondamentale nella realizzazione teorica del funzionamento di un elaboratore.

I primi elaboratori erano macchine enormi, costruite con componenti estremamente ingombranti e con consumi di energia notevolissimi.

ENIAC fu pubblicizzato come giant brain, origine del demenziale termine di cervello elettronico (o, confidenzialmente "cervellone" che ancora oggi, incredibilmente, si sente).

Nella realtà, la tecnologia degli elaboratori ha due componenti fondamentali:

- l'hardware, cioè l'insieme dei componenti fisici dell'elaboratore, utilizzati come base per i calcoli;
- il software, cioè l'insieme delle istruzioni che devo essere eseguite per risolvere qualsiasi tipo di problema si sottoponga all'elaboratore.

Senza il software, l'elaboratore potrebbe essere identificato come un "veloce citrullo" più che un cervello elettronico; è il software che dà l'intelligenza necessaria per raggiungere i risultati richiesti dal punto di vista logico.

Mentre nei primi anni l'elaboratore eseguiva le operazioni richieste attraverso manipolazioni analogiche delle sue componenti al fine di raggiungere i risultati e quindi il software (nel senso moderno) era praticamente inesistente, nel seguito furono sviluppati degli appositi strumenti tecnici per permettere la creazione del software: i linguaggi di programmazione.

Questo tipo di sviluppo fu fondamentale per la diffusione dell'elaboratore, dato che ne permetteva l'utilizzo anche a coloro che ne ignoravano i tecnicismi: era nato l'elaboratore nel senso moderno del termine.

I primi linguaggi di programmazione sono linguaggi "uno a uno" cioè linguaggi in cui un'istruzione software corrispondeva ad una sola istruzione all'hardware.

Ovviamente, questi linguaggi non erano di facile utilizzo, dato che richiedevano, se non una conoscenza approfondita del funzionamento dell'hardware, almeno una comprensione della sua struttura e del suo funzionamento.

Il vero salto verso la diffusione dell'elaboratore fu la realizzazione dei linguaggi "uno a molti" o di alto livello, dato che ad una sola istruzione scritta dal programmatore venivano eseguite molte istruzioni nell'hardware.

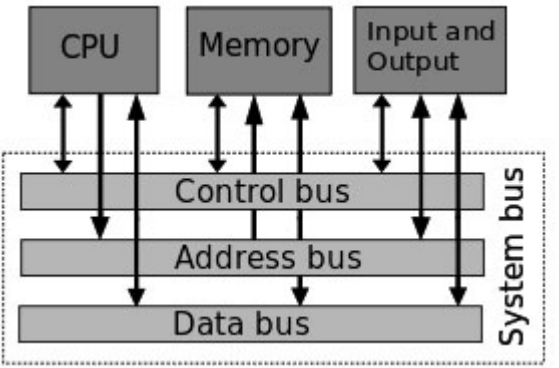
Con questo approccio, una istruzione "nascondeva" i dettagli tecnici dell'hardware, permettendo l'avvicinamento all'utilizzo dell'elaboratore da parte di persone orientate alla risoluzione del problema e non al funzionamento della tecnologia.

I primi linguaggi a diffondersi largamente furono orientatati a problemi scientifici (il FORTRAN – FORmula TRANslaction nel '57) e subito dopo orientati a problemi economici (COBOL – COmon Business Oriented Language nel '59).

Accanto a questi linguaggi, che divennero subito popolari, anche altri linguaggi specializzati furono realizzati per affrontare problemi più specifici.

Questi primi linguaggi erano orientati ad un modello hardware ben preciso, noto come modello di Von Neumann, il cui schema è riportato nella figura 1.

Fig. 1 – Modello di Von Neumann



Fonte: http://it.wikipedia.org/wiki/File:Computer_system_bus.svg.

Lo schema si basa su cinque componenti fondamentali:

1. **CPU** – Central Processing Unit – (o unità di lavoro) che si divide a sua volta in:
   1. unità operativa, nella quale uno dei sottosistemi più rilevanti è l'unità aritmetica e logica (o ALU);
   2. unità di controllo (CU);
2. unità di memoria, intesa come memoria di lavoro o memoria principale (**RAM** – Random Access Memory);
3. unità di input, tramite la quale i dati vengono inseriti nell'elaboratore per essere trattati;
4. unità di output, necessaria affinché i dati elaborati possano essere restituiti all'operatore;
5. bus, canali di collegamento di tutte le componenti fra loro.

L'esistenza dei linguaggi di alto livello fu ovviamente una condizione importante per aprire l'utilizzo degli elaboratori a studiosi di discipline diverse da quelle tipicamente scientifiche o aziendali.

Il contributo di Von Neumann fu fondamentale per lo sviluppo dell'informatica e della matematica applicata: infatti negli anni in cui l'informatica si consolidava, Von Neumann mise a punto anche la teoria dei giochi (una modellizzazione matematica delle scelte ottimali fra due decisori antagonisti) e soprattutto contribuì alla nascita e lo sviluppo del metodo di Montecarlo.

È ovvio che l'elaboratore sia uno strumento di tipo assolutamente deterministico, dato che opera su specifiche del suo programma. Tuttavia, in alcuni casi è necessario disporre di dati casuali per la soluzione di problemi di tipo probabilistico: occorre quindi disporre di numeri generati casualmente in relazione a determinate distribuzioni di probabilità ed introdurli nell'elaboratore per la realizzazione delle applicazione che li richiedono.

La quantità di dati di questo tipo ed il loro reperimento sarebbe un grosso deterrente all'utilizzo e allo sviluppo di applicazioni basate su distribuzioni di probabilità.

Il metodo di Montecarlo (che evoca proprio nel suo nome l'alea associata alla roulette e quindi al calcolo delle probabilità) permette di risolvere il problema: i valori numerici casuali vengono generati direttamente dall'elaboratore e permettono di trattare le distribuzioni di probabilità richieste.

Dato che la generazione di questi numeri avviene mediante un opportuno programma e non mediante estrazioni fisiche, normalmente tali numeri si dicono pseudo-casuali, proprio per distinguerli da quelli originali, ma hanno le stesse caratteristiche degli originali numeri casuali, permettendo all'elaboratore di affrontare non solo problemi deterministici, ma anche problemi soggetti a variabili aleatorie.

La disponibilità di strumenti di questo genere ha permesso di sfruttare la potenza di calcolo dell'elaboratore per generare una nuova area applicativa: la **simulazione probabilistica**.

Mediante questo strumento è possibile valutare scenari operativi soggetti a distribuzioni di probabilità sfruttando la potenza di calcolo offerta dall'elaboratore.

Data l'importanza del problema, sono stati sviluppati specifici linguaggi per facilitare la realizzazione di applicazioni orientate alla simulazione.

Per tutti, citiamo SIMULA, un linguaggio di simulazione sviluppato da Kristen Nygaard (un ricercatore operativo) e Ole-Johan Dahl (un esperto di informatica) all'Università di Oslo negli anni '60.

L'obiettivo del loro lavoro era quello di creare uno strumento che permettesse la simulazione di eventi discreti e, nel loro specifico caso, di analizzare l'andamento delle maree nei fiordi norvegesi.

Poiché esisteva ALGOL (Algorithmic Language), un ambiente particolarmente adatto a rappresentare un efficiente supporto per il progetto, SIMULA si appoggiò a questo linguaggio, diventato il primo strumento adatto ad affrontare problemi di simulazione numerica.

Abbiamo parlato specificamente di SIMULA perché, come vedremo, questo linguaggio fu la base per creare negli anni '80 un nuovo modello di elaborazione di cui parleremo diffusamente: la programmazione ad oggetti.

Man mano lo sviluppo dell'informatica progrediva, aumentavano le aree in cui l'elaboratore veniva applicato ed in particolare si cominciava a parlare dell'intelligenza artificiale[2].

2. A parte la difficoltà di definire in modo completo e corretto cosa sia l'intelligenza naturale (il che rende oltremodo difficile capire cosa sia l'intelligenza artificiale), sono personalmente convinto che il termine indichi una forma pericolosa di vapor-ware cioè discussioni sul nulla, dato che in 50 anni di lavoro nel settore informatico, non mi è mai capitato di vedere qualcosa che sia lontanamente capace di essere avvicinato al pensiero umano; per chi fosse interessato all'argomento suggerisco la lettura di *Demystifying Machine Intellgence* di Piero Scaruffi, sulle cui posizioni sono totalmente d'accordo; se poi volesse approfondire le idee

Con questo termine si intende l'abilità dell'elaboratore a svolgere ragionamenti e attività tipiche della mente umana.

Il primo linguaggio specializzato per questo tipo di attività fu il LISP, nato più o meno nello stesso periodo del FORTRAN ma molto meno diffuso.

Mentre si sviluppavano tutte queste aree di ricerca, il mercato dell'elaborazione era in mano completamente a IBM, che faceva la parte del leone, mentre altre aziende (i cosiddetti sette nani: Burroughs, Control Data, General Electric, Honeywell, NCR, RCA e UNIVAC) si muovevano a loro volta per crearsi nicchie di mercato.

In particolare General Electric aveva sviluppato sistemi time-sharing con la collaborazione del Dartmouth College, uno delle più prestigiose università americane che faceva parte della cosiddetta ivy league.

Il time-sharing era un tipo di elaborazione molto simile a quello che oggi è Internet, seppure su una dimensione enormemente più limitata e del tutto differente dall'approccio di GE.

Il Dartmouth College è un'entità di primo piano nella storia dell'informatica, dato che due suoi professori (John George Kemeny e Thomas Eugene Kurtz) svilupparono il Basic (Beginner's All-purpose Symbolic Instruction Code).

Il Basic, inizialmente creato come linguaggio compilato, divenne in seguito il primo programma interpretato[3] e fu inizialmente sviluppato su un elaboratore GE.

Grazie ai due precedenti scienziati, venne realizzato anche il primo time-sharing, sempre sulle macchine GE.

Vediamo di capire cos'è il time-sharing (suddivisione del tempo).

dell'intelligenza artificiale di Alan Turing, suggerisco il recente libro pubblicato dal Corriere della Sera *Turing. La nascita dell'intelligenza artificiale'* curato da Mattia Monga.

3. Il Basic aveva elementi derivati sia dal Fortran che dall'Algol, che lo avevano preceduto e che erano programmi *compilati* (cioè ottenuti attraverso l'utilizzo di un altro programma che generava il cosidetto *eseguibile*, leggendo e convertendo il programma scritto dal programmatore umano).

In un programma *interpretato* questo passaggio è evitato; lo svantaggio è la maggior lentezza dell'esecuzione (anche se, con i moderni elaboratori, è pressoché irrilevante).

Uno delle sue istruzioni (GOTO) fu nel seguito duramente criticato da Dijkstra, che lo individuò (a ragione) come il generatore dei programmi spaghetti-style (cioè programmi estremamente difficili da manutenere), proponendo come alternativa la programmazione strutturata, che si contrapponeva alla programmazione procedurale.

Nella programmazione procedurale è consentito il comando GOTO, cioè un "salto" da un'istruzione all'altra all'interno del programma, rendendo difficile seguire il flusso dell'elaborazione ed in conseguenza la verifica del suo corretto funzionamento.

Nel 1966 il teorema di Böhm-Jacopini dimostrò che ogni programma poteva essere realizzato secondo 3 sole strutture: la sequenza (una istruzione dopo l'altra), la selezione (la scelta tra più cammini differenti) ed il ciclo (o iterazione), da applicare ricorsivamente alla composizione di istruzioni elementari; l'utilizzo di questo approccio migliorò enormemente la manutenibilità dei programmi (elemento fondamentale per il loro utilizzo).

Come detto precedentemente, il tipo di **elaborazione batch** (cioè un'elaborazione basata sull'elaborazione sequenziale dei programmi da elaborare) era, in pratica, l'unico approccio all'elaborazione dati dei primi anni '60; in questo tipo di elaborazione (ricordiamo: molto costosa), il tempo di utilizzo era condizionato dall'immissione dei dati, che non era proprio velocissima, trattandosi di schede o di supporti magnetici che dovevano essere montati secondo le necessità.

Organizzando opportunamente lo sfruttamento delle risorse veloci di elaborazione, il lavoro poteva essere ripartito fra vari utilizzatori (**multiprogrammazione** o **multitasking**), passando rapidamente fra i vari lavori richiesti e consentendo quindi all'elaboratore di svolgere diversi lavori "contemporaneamente".

Queste condizioni, non erano da sole sufficienti a garantire un soddisfacente servizio dato che, ad esempio, un certo lavoro poteva durare molto a lungo, penalizzando il lavoro successivo.

Si trattava quindi di realizzare un sistema di interruzioni hardware in grado di interrompere un lavoro in corso di esecuzione per dare spazio al successivo che a sua volta lavorava per un intervallo di tempo prefissato (ecco la ragione del nome); alla fine dell'intervallo di tempo, il controllo ritornava ai lavori in attesa.

Fu così realizzato un sistema di elaborazione che aveva un grandissimo vantaggio rispetto alle elaborazioni batch, non solo dal punto di vista economico, ma anche dal punto di vista dell'efficienza.

Grazie a questi sviluppi nell'elaborazione, l'intelligenza artificiale divenne un punto di grande interesse.

Secondo Marvin Minsky (uno dei pionieri e dei massimi esponenti di quest'area dell'informatica), l'intelligenza artificiale vuol dire "far fare alle macchine delle cose che richiederebbero l'intelligenza se fossero fatte dagli uomini".

Già nel '50 Alan Turing (quindi ben prima che un "vero" elaboratore cominciasse a funzionare) si era posto il quesito se l'elaboratore avrebbe potuto "comportarsi" come un uomo e aveva ipotizzato che per il 2000 questa ipotesi avrebbe potuto essere verificata; ad oggi, nonostante gli enormi miglioramenti della tecnologia informatica, non ci sono ancora certezze (anzi, molti dubbi).

Turing ipotizzò un gioco di imitazione, che viene chiamato il *test di Turing*, strutturato come nella figura 2.

Supponiamo che ci siano 3 persone: Alberto, Beatrice e Caio; Caio riceve delle risposte (scritte, per evitare ogni indizio, per identificare Alberto e Beatrice) e deve capire chi è l'uomo e chi è la donna.

Alberto ha il compito di ingannare Caio, mentre Beatrice ha il compito di aiutarlo a trovare la risposta giusta.

Nel test, si ipotizza che Alberto sia sostituito da un elaboratore; se dopo la sostituzione, Caio mantiene la percentuale di valutazioni corrette, possiamo pensare che l'elaboratore che risponde invece di Alberto sia "intelligente".