AI: il test di Turing e gli algoritmi minimax

Ester Dalvit

10 marzo 2005

1 Intelligenza artificiale

L'intelligenza artificiale (AI, da Artificial Intelligence) è la scienza che studia come far fare alle macchine cose che, se fatte dagli umani, richiedono intelligenza. Può essere anche definita come lo studio dei processi che rendono possibile percepire, ragonare ed agire.

Dopo una fase di ottimismo negli anni '50 e '60, ci si è resi conto, anche a causa di molti insuccessi, che far fare alle macchine ciò che gli esseri umani sanno fare è veramente difficile.

Infatti una macchina compie in modo efficiente e veloce attività come

- eseguire calcoli aritmetici
- manipolare stringhe (es: cercare una parola in un testo)
- eseguire processi ripetitivi (es: fare migliaia di volte la stessa operazione)
- obbedire esattamente ai comandi.

Invece alcune abilità proprie degli esseri umani, che potremmo definire intelligenza, sono:

- percepire, ovvero analizzare l'ambiente con organi di senso (naturali o artificiali) individuando oggetti e relazioni tra essi;
- agire (camminare, raccogliere oggetti, usare attrezzi, ...);
- comprendere il linguaggio naturale e generare frasi di senso compiuto;
- imparare (dagli esempi, per tentativi, attraverso l'osservazione, ...);
- ragionare, avere buon senso, cioè derivare conseguenze da una situazione (con metodo induttivo o deduttivo) che siano utili e rilevanti alla situazione corrente o all'obiettivo da perseguire;
- giocare (scacchi, parole crociate, ...);
- fare matematica e logica formale: dimostrare teoremi, risolvere problemi (partendo dai certi dati, trovare qualcosa di determinato);

- analizzare situazioni e progettare (ingegneria, architettura, diagnosi mediche, avvocati, pianificazione finanziaria, ...).

Spesso gli obbiettivi più facili per l'uomo sono i più difficili da far perseguire alle macchine.

I temi principali di cui si occupa l'intelligenza artificiale sono:

- machine learning: far imparare alle macchine a partire dall'esperienza
- ragionamento automatizzato
- pattern recognition
- planning/scheduling: pianificazione
- natural language processing: comprensione e interazione
- game playing
- dimostrazione di teoremi, logica
- computer vision
- percezione/azione
- robotica
- rappresentazione del sapere
- sistemi basati sulla conoscenza di campi specifici e delle regole da applicare (sistemi esperti)
- tecniche di ricerca
- architettura dei computer per l'intelligenza artificiale
- applicazioni e implicazioni sociali

Alcuni campi di applicazione dell'intelligenza artificiale sono il commercio, dove può essere utile ad esempio programmare sistemi che aiutano a fare l'inventario o che pianificano il lavoro, e il design, dove si possono pensare programmi per criticare i progetti o identificare i rischi.

Si distingue l'intelligenza artificiale "debole" da quella "forte": la prima ha come obiettivo far simulare alle macchine l'intelligenza umana; la seconda ritiene possibili che le macchine stesse siano intelligenti.

2 Il Test di Turing

Nell'articolo *Computing machinery and intelligence* (1950), Alan Turing propone di considerare la domanda "Le macchine sono in grado di pensare?"; egli nota che la domanda è mal posta ed ambigua: sarebbe necessario definire che cos'è una macchina e che cosa significa pensare.

Invece di tentare di rispondere a questa domanda, Turing propone di sostituirla con un'altra, che descrive un modo operativo di procedere. Questo sarà chiamato test di Turing.

Esso verifica se una macchina ha la capacità di conversazione uguale a quella di un uomo. Non è un test d'intelligenza vero e proprio; testa piuttosto fino a che punto un computer sia in grado di simulare una conversazione umana.

Il test di Turing così come l'autore lo descrisse non è mai stato superato da alcuna macchina, ma limitando gli argomenti della conversazione e restringendo il test, si sono avuti dei tentativi coronati da successo.

2.1 L'*Imitation Game*

Il test fu ispirato da un gioco spesso proposto nelle feste fra amici, detto *Imitation Game*. In questo gioco un uomo e una donna entrano in due stanze separate, mentre gli altri giocatori, che non sanno in quale delle due stanze sia la donna e in quale l'uomo, hanno il ruolo di giudici.

Essi rimangono fuori dalle stanze e pongono delle domande ai due, che scrivono le risposte su dei fogli. I giudici, leggendo le risposte, devono capire chi dei due è la donna.

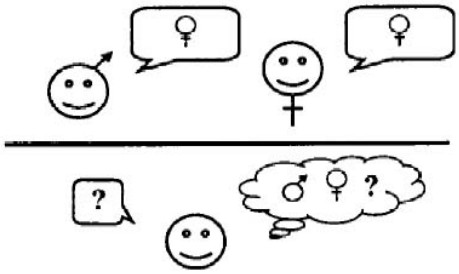


Figura 1: Imitation game

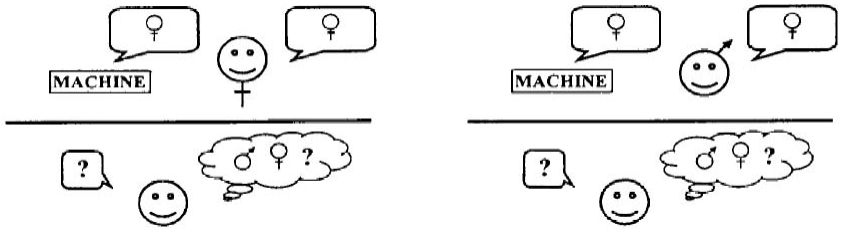
L'uomo e la donna rinchiusi cercano entrambi di comportarsi come donne: l'uomo vince se fa credere ai giudici di essere una donna; la donna vince se i giudici fanno l'identificazione corretta.

Lo schema del gioco è illustrato nella figura 1.

Per evitare possibili identificazioni basate sulla calligrafia, si può decidere di affidare a una persona neutrale la lettura ai giudici delle risposte scritte dai due concorrenti.

Ora immaginiamo che la parte dell'uomo o della donna sia presa da una macchina, che deve comportarsi come la persona di cui ha preso il posto.

Nella figura 2 è illustrato questo passo intermedio per arrivare al test di Turing.



(a) La macchina al posto dell'uomo (b) La macchina al posto della donna

Figura 2: Passo intermedio

A questo punto possiamo sostituire la pretesa di essere donna, sostenuta in questo passo intermedio da una macchina e una persona, con quella di essere un umano (figura 3).

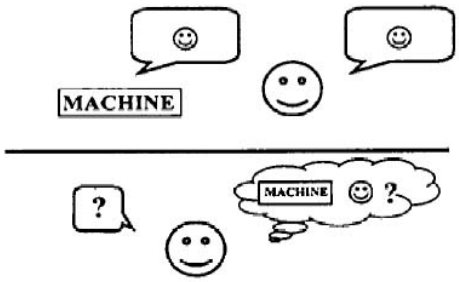


Figura 3: Il test di Turing

Il test di Turing può essere descritto in questo modo: una persona, che funge da giudice, intraprende una conversazione sia con un'altra persona che con una macchina; entrambe devono avere un comportamento umano.

La conversazione avviene solo in forma scritta, in modo che il test per valutare la capacità linguistica di una macchina sia semplice ed universale.

Il giudice può porre qualsiasi tipo di domanda, e il computer può fare qualsiasi cosa per trarlo in inganno: ad esempio può rispondere negativamente alla domanda "sei un computer?", oppure se il giudice chiede di svolgere un calcolo numerico, la macchina può rispondere dopo una lunga pausa e con qualche errore.

L'uomo interrrogato deve cercare di aiutare il giudice a fare la corretta identificazione.

Se il giudice non riesce a stabilire quale sia la persona e quale la macchina, la macchina supera il test. Naturalmente il giudice non deve tirare a indovinare, ma basarsi su prove concrete, fornite nelle risposte.

2.2 Obiezioni

Lo stesso Turing nel suo articolo propose delle obiezioni al suo test, e le confutò. Eccone alcune:

- **Obiezione Teologica:** pensare è una funzione dell'anima umana, quindi una macchina non può pensare.
  Replica: Dio può dotare anche le macchine di un'anima, se vuole.

- **Obiezione Matematica:** si basa su teoremi matematici, come il Teorema di incompletezza di Gödel, per dimostrare che esistono domande a cui non si può rispondere basandosi solo sulla logica, come fanno le macchine.
  Replica: ogni macchina ha delle domande diverse a cui non può rispondere, non esiste una domanda a cui nessuna macchina sappia rispondere; inoltre anche gli umani spesso sbagliano o non sanno rispondere.

- **Coscienza:** una macchina non può dirsi intelligente finchè non scrive poesie o canzoni, ispirata da emozioni provate, e finchè non è cosciente di averle scritte.
  Replica: non si può dire che un uomo sia intelligente, l'unico modo per sapere che egli pensa è essere quell'uomo; solitamente si adotta la convenzione che ognuno pensi. Inoltre durante una conversazione ci si può rendere conto se l'interlocutore sa di cosa stiamo parlando dalle sue risposte; se una macchina, in un gioco come l'Imitation Game, dà risposte che sembrano ragionevoli, siamo indotti a credere che sia una persona. Dunque dobbiamo accettare il test.

- **Varie disabilità:** le macchine non sanno essere simpatiche o belle, e così via, inoltre non sbagliano mai.
  Replica: siamo indotti a pensarlo per induzione, perchè conosciamo solo macchine che hanno queste caratteristiche; solo le macchine ideali non sbagliano.

- **Obiezione di Lady Lovelace:** le macchine non fanno nulla di originale, fanno solo ciò che noi ordiniamo, seguendo schemi predeterminati.
  Replica: quale uomo può dire di creare qualcosa dal nulla? quello che fa può essere il risultato di ciò che ha imparato, anche se non se ne rende conto. Le macchine possono stupire l'uomo, specialmente quando le conseguenze dei fatti non sono riconoscibili immediatamente.

- **Informalità del comportamento:** non è possibile descrivere un insieme di regole esaustivo per l'uomo, ovvero che dica come un uomo si comporta in ogni possibile situazione. Un sistema governato da leggi è prevedibile e quindi non intelligente.
  Replica: questo dimostra che un uomo non può essere una macchina. Inoltre si stanno confondendo le leggi di comportamento (fisiche) con le regole generali di condotta (cosa bisogna fare).

- **Percezioni extrasensoriali:** Turing crede che ci siano prove dell'esistenza di percezioni extrasensoriali. Queste potrebbero influenzare lo svolgimento del test.
  Replica: si possono creare condizioni per cui queste perczioni non influiscano sull'andamento del test.

In seguito si sono trovate delle argomentazioni per mostrare che il test di Turing non è una valida definizione dell'intelligenza delle macchine: ne descriviamo alcune.

Innanzitutto una macchina potrebbe essere molto intelligente, senza per questo essere capace di parlare come una persona.

In secondo luogo molte persone che vorremmo considerare intelligenti potrebbero fallire il test (ad esempio, i bambini o gli analfabeti).

Infine una macchina che superi il test di Turing è capace di simulare un comportamento umano durante una conversazione, ma ciò non equivale ad essere intelligente. La macchina infatti può semplicemente seguire delle regole predeterminate, scelte in modo intelligente (un'obiezione a questo è la domanda: come facciamo ad essere sicuri che anche gli esseri umani non seguano delle regole predeterminate?). Due esempi famosi di questa argomentazione sono la "Chinese room" e la "Blockhead".

La stanza cinese è un'obiezione sollevata da John Searle nel 1980, contro una teoria diffusa nell'ambito dell'intelligenza artificiale: se una macchina supera il test di Turing, allora si può ritenere che essa sia intelligente come un essere umano. Searle ritiene invece che se anche una macchina riuscisse a superare il test di Turing, ciò non assicurerebbe affatto che essa abbia un'intelligenza cosciente.

Immaginiamo che al posto della macchina ci sia un uomo, rinchiuso in una stanza, che non conosce il cinese, e anzi non dà alcun significato ad alcuno dei caratteri cinesi. Egli ha a disposizione un libro di regole scritte in inglese, lingua che egli comprende (questo fa la parte dei programmi), dei fogli su cui sono scritti caratteri cinesi, che corrispondono al database, e altri fogli bianchi su cui può scrivere, che simboleggiano la capacità di calcolo della macchina.

Le regole spiegano come mettere in relazione un insieme di simboli formali con un altro insieme di simboli formali: "formali" significa che si possono identificare per la loro forma.

A questa persona vengono passati dei fogli con dei simboli, che per le persone esterne sono una storia e delle domande. Egli manipola i simboli che gli vengono dati seguendo le regole scritte nel libro e scrive su un foglio dei caratteri, che le persone all'esterno chiameranno risposta.

Se le regole sono scritte in modo intelligente, l'uomo passa il test di Turing, perchè risponde correttamente in cinese alle domande poste in cinese. Ciò significa che un cinese che leggesse domande e risposte, cioè interpretasse i caratteri che l'uomo nella stanza scrive, le troverebbe coerenti, ovvero indistinguibili dalle risposte che darebbe un cinese, dunque sarebbe indotto a credere che la persona che risponde conosce il cinese. Tuttavia l'uomo rinchiuso nella stanza non capisce il significato di ciò che gli viene chiesto nè di ciò che egli stesso risponde. Egli segue semplicemente delle regole; si comporta esattamente come un computer.

Non si può dunque dire che una macchina che superi il test di Turing sia in grado di comprendere input e output: l'uomo nella stanza cinese potrebbe perfino non essere cosciente di rispondere a delle domande.

Le premesse di questo argomento sono: le macchine si basano sulla forma (sintassi), mentre la mente umana dà un significato ai simboli (semantica); inoltre la sintassi da sola non è sufficiente per costruire la semantica. Queste

ipotesi portano alla conclusione che le macchine non sono in grado di ragionare in termini di semantica, cioè si basano su simboli a cui non danno alcun significato.

Il filosofo Ned Block descrive nell'articolo "Psychologism and Behaviourism" una macchina teorica, chiamata Blockhead. Egli vuole dimostrare che possono esistere macchine non intelligenti che superano il test di Turing: Blockhead ha queste proprietà.

Block descrive una conversazione che duri un tempo limitato. Egli ritiene che ci siano un numero finito di frasi corrette dal punto di vista sintattico e grammaticale che possano essere usate per iniziare una conversazione. Da ciò segue che il anche il numero di risposte corrette a questa prima frase sia finito, e ciò vale anche per tutte le risposte seguenti.

Ora immaginiamo una macchina programmata con tutte queste frasi possibili: sebbene il numero di frasi necessarie per una conversazione anche breve sia enorme, una macchina di questo tipo può esistere almeno teoricamente. Allora essa può continuare una conversazione con una persona su qualsiasi argomento, perchè è programmata con ogni frase possibile. In questo modo, la macchina può superare il test di Turing, anche se non è intelligente.

3 Alcuni programmi

In questa sezione presenteremo alcuni programmi che tentano di dialogare con l'utente, assumendo un comportamento umano, per superare il test di Turing: tali sistemi richiedono conoscenze di *natural language processing*; vediamo allora alcuni problemi del linguaggio naturale e come essi possono essere risolti.

3.1 Il linguaggio naturale

Come abbiamo visto, per parlare di intelligenza artificiale, abbiamo bisogno di sapere cos'è e cosa fa l'intelligenza. Nello stesso modo dobbiamo partire dall'essere umano, per studiare come trattare il linguaggio.

Il linguaggio naturale è ciò che gli esseri umani usano per comunicare, principalmente le varie lingue, parlate e scritte. Esso è costituito da un insieme di simboli che hanno un significato convenzionale.

Questa breve introduzione ai problemi che possono sorgere dall'interpretazione del linguaggio naturale sarà ripresa quando analizzeremo qualche esempio di come una macchina può trattare il linguaggio naturale, problema studiato dal *Natural Language Processing (NLP)*.

Analizziamo con alcuni esempi il modo con cui noi interpretiamo il linguaggio. Questi stessi schemi, che noi usiamo inconsciamente, si dovranno applicare alle macchine.

Ad esempio, dato l'input:

```
Anna e Bruno andarono al ristorante.
Dopo essersi seduti, ordinarono da mangiare.
Più tardi pagarono il conto e tornarono a casa.
```

ci si potrebbe chiedere se Anna e Bruno abbiano mangiato qualcosa. Il testo non lo dice esplicitamente, ma il buon senso lascia supporre di sì.

Da qui in avanti gli esempi saranno in inglese, perchè questo è il linguaggio naturale solitamente usato nell'interfaccia utente-macchina. Si potrebbe fare l'analisi anche per la lingua italiana: emergerebbero schemi diversi e altre forme espressive.

Alcune difficoltà che si possono incontrare nel *NLP* sono esemplificate nelle frasi seguenti.

```
John drove his sister to buy groceries.
John drove his car to buy groceries.
John drove his sister to commit suicide.
John drove his car to commit suicide.
```

La difficoltà consiste nell'uso delle stesse strutture per indicare azioni diverse. Un essere umano che conosca l'inglese riesce a dare il giusto significato a queste frasi, perchè, considerando i primi due periodi, sa distinguere il significato di "drove his car", guidò la sua automobile, da quello di "drove his sister", accompagnò sua sorella con l'automobile. Per distinguere i due casi bisogna sapere che "car" è un oggetto, mentre "sister" indica una persona, e che il verbo "drive" assume significati diversi a seconda della natura del complemento oggetto.

Nel terzo periodo il significato di "drive" è ancora diverso: John non accompagna in automobile la sorella, ma la induce al suicidio. Per capire il senso di

questa affermazione bisogna valutare, oltre al complemento oggetto, anche la frase finale, ma dal punto di vista della struttura ("to" + verbo + complemento oggetto) essa è perfettamente identica alla frase finale del primo periodo.

Ecco un altro esempio di affermazione che noi capiamo ma che costituiscono una difficoltà per il *NLP*

```
My thoughts turn on the past.
```

Leggendo scartiamo subito la possibile lettura

My thoughts turn on the past.

ovvero "I miei pensieri accendono il passato", e adottiamo

My thoughts turn on the past.

che significa "I miei pensieri tornano al passato". In questo caso la difficoltà è data dall'uso della stessa forma per esprimere significati diversi. In un caso consideriamo come forma verbale "turn on" e come come complemento oggetto quello che segue, mentre nell'altro caso consideriamo "turn" come verbo e "on the past" diventa complemento di termine.

In alcuni casi la stessa frase può essere ambigua, ad esempio

```
I turn on the radio.
```

può essere interpretata in modi diversi a seconda del contesto. Essa può significare sia "Accendo la radio":

```
I need to listen to music. I turn on the radio.
```

che "Scelgo la radio":

```
I want to buy a present for her, a mobile phone or a radio.
I don't think she would like a mobile phone.
I turn on the radio.
```

3.2 Il *Natural Language Processing*

Il *Natural Language Processing* cerca di risolvere i problemi che abbiamo appena visto, per fare in modo che un computer riesca a simulare la comprensione del linguaggio naturale e ad elaborare risposte appropriate.

Per costruire una macchina che passi il test di Turing è necessario fare in modo che riconosca:

- le parti del linguaggio (nomi, verbi, preposizioni, ecc.)
- la sintassi, per cui deve scartare la frase "Boy the the store to go", anche se tutte le parole sono corrette
- la semantica, per cui ad esempio la frase "Colorless green ideas sleep furiously" è priva di significato
- concetti sottintesi (pragmatica): ad esempio "Do you know what time it is?" implica "Please tell me what time it is"

Dobbiamo quindi definire una grammatica:

- un insieme di simboli costituiti da lettere (parole in un linguaggio naturale)
- un insieme di simboli per i componenti sintattici
- un insieme di regole di riscrittura
- un simbolo di start
- una rappresentazione delle parole.

Un esempio di una semplice grammatica:

S	→	NP VP
NP	→	[DET] [ADJ]* NOUN [PP]*
VP	→	VERB [NP] [PP]*
PP	→	PREP NP
NOUN	→	Jane
NOUN	→	Bill
NOUN	→	book
VERB	→	sold
DET	→	a
PREP	→	to

ove [ ] indica un argomento opzionale, e [ ]* indica un argomento ripetuto zero o più volte.

Lo schema di quello che accade è il seguente:

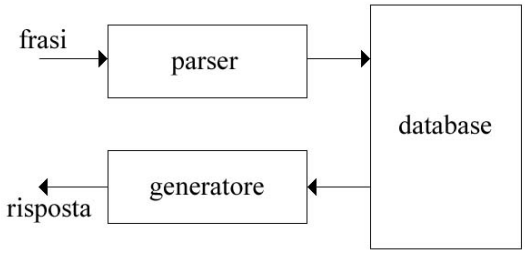


Figura 4: Parser

Il *parsing* è il processo della trasformazione di una frase in linguaggio naturale in una forma manipolabile dalla macchina.

Le frasi in linguaggio naturale vengono tradotte in asserzioni logiche dal parser e memorizzate o confrontate nel database. Una domanda in linguaggio naturale viene tradotta dal parser in una ricerca nel database.

Perchè non usare l'inglese per la rappresentazione delle frasi nel database? Rispondiamo con un esempio. Consideriamo la frase "Jane sold a book to Bill", che costituisce l'input. Allora il suo significato per la macchina sarà `(Jane sold a book to Bill)`; memorizziamolo nel database.

Ora proviamo a porre delle domande, a cui la macchina possa rispondere sì o no. Scriviamo l'input: "Has Jane sold a book to Bill?", che il parsec

tradurrà in `query (Jane sold a book to Bill)`. La query indica che bisogna effettuare una ricerca. L'affermazione si trova nel database, la risposta sarà quindi affermativa.

Finora va tutto bene. Consideriamo altre domande: "Has anyone sold a book to Bill" oppure "Who sold a book to Bill?". Attraverso il parsing verranno trasformate in `(query ((var1) sold a book to Bill))`. Effettuando la ricerca nel database, se troviamo `(Jane sold a book to Bill)`, avremo che (var1) assume il valore "Jane" e potremo rispondere.

Ma se la domanda viene formulata in modo leggermente diverso, come "Did anyone sell a book to Bill?", essa verrà trasformata in `(query ((var1) sell a book to Bill))` ed effettuando questa ricerca non troviamo la frase `(Jane sold a book to Bill)`, perchè il verbo è diverso.

Dovremo quindi distinguere i tempi dei verbi: per indicare "sold", "did sell", "has sold" useremo `(sell past)`. In questo modo domande che hanno lo stesso significato, come "Did anyone sell a book to Bill?" e "Has anyone sold a book to Bill?", generano entrambe la `(query ((var1) (sell past) a book to Bill))`.

Consideriamo altri problemi: possono venir poste altre domande, come "Was a book sold to Bill by Jane?", "Was Bill sold a book by Jane?" e "Was Bill sold a book?". La cosa più semplice è di non memorizzare nel database asserzioni che contengano verbi al passivo.

Bisogna quindi trasformare la rappresentazione di "Bill was sold a book by Jane." in `((var1) (sell past) (var2) a book )`, dove `(var1)` indica il soggetto e `(var2)` il complemento di termine.

Possono esserci altri elementi nella frase, ad esempio si può specificare dove, o quando. Per memorizzare le frasi nel database bisogna continuare a scomporle.

Altri problemi possono derivare da affermazioni equivalenti, quali "the marble statue" e "the statue of marble", forma che non è permessa in tutti i casi: ad esempio "the brown horse" è corretto, ma "the horse of brown" non lo è.

Ritorniamo allora alla nostra grammatica:

soggetto	S	→	NP VP — VP
compl.oggetto	NP	→	[DET] [ADJ]* NOUN [PP]*
predicato	VP	→	VERB [NP] [PP]*
complementi	PP	→	PREP NP
	NOUN	→	Jane
	NOUN	→	Bill
	NOUN	→	book
	VERB	→	sold
	DET	→	a
	PREP	→	to

ove [ ] indica un argomento opzionale, e [ ]$*$ indica un argomento ripetuto zero o più volte.

Ora la frase "Jane sold a book to Bill" ha come significato:

```
( S (NP (NOUN Jane))
  (VP (VERB sold) (NP (DET a) (NOUN book))
  (PP (PREP to) (NP (NOUN Bill)))) )
```

Il parsing basato solo sulla sintassi può però risultare ambiguo. Per risolvere il problema utilizziamo anche la semantica.

La rappresentazione che descriviamo si chiama *case frame*: essa associa ad ogni verbo un insieme di casi (*case slots*), alcuni necessari, altri opzionali.

La parte centrale di questa rappresentazione è il verbo. Ogni *case frame* ha un insieme finito di *case slots* possibili (di solito tra 4 e 30).

In questo modo le frasi "Jane sold a book to Bill", "A book was sold by Jane to Bill", "Bill was sold a book by Jane" e "A book was sold to Bill by Jane" avranno tutte lo stesso significato

```
( sell (agent          Jane)
       (object         a book)
       (counter-agent  Bill)
       (tense          past))
```

Potremmo ora analizzare una frase come "Jane sold a copy of War and Peace to Bill", e memorizzarla nel database. Se poniamo la solita domanda "Did Jane sell a book to Bill?", il sistema risponderà negativamente.

Per rispondere correttamente il sistema NLP ha bisogno di ulteriore conoscenza: un modo per codificare la conoscenza della realtà (*world knowledge*) è l'uso di reti semantiche (*semantic nets*). Esse sono costituite da grafi, in cui i vertici indicano oggetti, classi e valori di attributi, mentre gli archi rappresentano tipi di attributo.

Con questa rappresentazione nascono altri problemi, ma qui non verranno analizzati.

Lo scopo di questa panoramica dei problemi del NLP è di far capire che noi ragioniamo secondo degli schemi, che spesso non riconosciamo mentre li applichiamo. Tuttavia per "insegnare" ad una macchina come interpretare il linguaggio naturale, è necessario studiare i nostri schemi di espressione, scomporre le affermazioni e tradurle in una forma manipolabile dalla macchina. Questo processo, come si è visto, è detto *parsing*.

Le stesse regole vanno applicate al contrario per generare linguaggio, ovvero per costruire frasi in linguaggio naturale che abbiano un senso e una sintassi corretta.

3.3 ELIZA

Vogliamo ora presentare ELIZA: scritto nel 1966 da Joseph Weizenbaum, che lo definì "un programma per computer per lo studio della comunicazione in linguaggio naturale tra uomo e macchina", esso è il primo tentativo compiuto per superare il test di Turing.

I programmi come ELIZA vengono chiamati sistemi di comprensione/generazione di linguaggio, o agenti di conversazione o semplicemente chatbots o chatterbot.

ELIZA emula una conversazione iniziale con uno psicoterapeuta Rogeriano: l'utente "dialoga" con il programma, che fa la parte dello psicologo. Il suo modo di dialogare è basato soprattutto sul riformulare le frasi dell'utente-paziente in domande da porgli e sull'invitarlo a continuare a parlare.

Per esempio, se il paziente scrive "I'm unhappy", la replica di ELIZA potrà essere "Why do you say you are unhappy?" oppure "Have you been unhappy often?" o ancora "What causes you to be unhappy?".

Weizenbaum disse che ELIZA è una parodia delle risposte di uno psicoterapeuta all'inizio di un'intervista psichiatrica. Egli scelse l'ambulatorio di uno psicologo per ambientare il suo programma, per evitare il problema di dover dotare il programma di una conoscenza del mondo reale. Infatti questo contesto è uno di pochi in cui un essere umano (e dunque il suo emulo) può replicare ad un'affermazione con una domanda che rivela la sua ignoranza sul tema discusso. Ad esempio, per la domanda "Qual è il tuo compositore preferito?", risposte accettabili in questo contesto sono "Dimmi invece quali compositori preferisci tu", oppure "Perchè ti interessa saperlo?".

Il *Natural Language Processing (NLP)* ha queste caratteristiche:

- l'input è costituito da frasi scritte in inglese (o qualche altro linguaggio naturale)
- il contenuto viene compreso
- le risposte sono in inglese (o comunque nella stessa lingua dell'input)

ELIZA è un programma per emulare il NLP; vediamo un esempio di conversazione tra utente e psicoterapeuta. Le frasi maiuscole sono scritte da 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
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
You are not very aggressive but I think you don't want me to
notice that.
WHAT MAKES YOU THINK I AM NOT VERY AGGRESSIVE
You don't argue with me.
WHY DO YOU THINK I DON'T ARGUE WITH YOU
You are afraid of me.
DOES IT PLEASE YOU TO BELIEVE I AM AFRAID OF YOU
My father is afraid of everybody.
WHAT ELSE COMES TO YOUR MIND WHEN YOU THINK OF YOUR FATHER
```

```
Bullies.
DOES THAT HAVE ANYTHING TO DO WITH THE FACT THAT YOUR
BOYFRIEND MADE YOU COME HERE
```

Cosa fa il programma? Cerca le parole chiave, riconosce gli schemi e usa semplici regole di riscrittura.

1. Identifica la parola chiave più importante all'interno di una frase
2. Individua un contesto minimo attorno alla parola chiave
3. Sceglie una regola di trasformazione appropriata
4. Trova cosa fare se nessuna delle regole di trasformazione è adatta

Ad esempio, data la frase in input

It seems that you hate me.

essa viene rielaborata:

$$\underbrace{\text{It seems that}}_{1} \quad \underbrace{\text{you}}_{2} \quad \underbrace{\text{hate}}_{3} \quad \underbrace{\text{me}}_{4}.$$

La ricerca di parole chiave avviene confrontando le parole della frase con una lista di parole "note", cioè presenti nel database. In questo esempio vengono individuate le parole chiave *you*, con rango 5, e *me*, con rango 3.

ELIZA trova la prima regola di decomposizione che si può applicare al caso. Uno schema che si può applicare a questa frase è: `(0 you 0 me)`, dove 0 indica un numero qualsiasi di parole.

Un altro schema adatto è `(0 you 1 me)`, ma esso non avrebbe funzionato per la frase "It seems that you love and hate me": le regole da applicare devono essere il più generale possibili, altrimenti servirebbero moltissime regole.

Successivamente "you" viene trasformato in "I" e "me" in "you".

La replica allora potrà essere "What makes you think I 3 you", dove 3 indica la parola numerata con questo numero, ovvero "hate". La ricomposizione avviene scegliendo lo schema usato meno recentemente.

ELIZA è un programma che simula intelligenza, in realtà non ha nessuna comprensione delle fasi scritte dal paziente. La sua forza consiste nel rispondere con frasi che sembrano ragionevoli la maggior parte delle volte.

3.4 PARRY

PARRY è un programma simile ad ELIZA: scritto da Colby nel 1970, simula una persona paranoica: ha tratto in inganno diversi psicologi, che, dialogando con lui, credevano di avere veramente a che fare con un paranoico.

Ecco un esempio di come risponde PARRY (frasi maiuscole).

```
Tell me something about yourself.
I WENT TO THE TRACK AT BAY MEADOWS A WHILE BACK.
What happened there?
THERE'S A LOT OF CROOKED GAMBLING GOING ON THERE.
Why do you think the gambling is crooked?
I HAVE WORRIED THAT GANGSTERS HAVE TOO MUCH POWER IN THIS
COUNTRY.
How do you like the hospital?
I DON'T COMMUNICATE MUCH WITH OTHERS.
```

A differenza di ELIZA, che "ascolta" l'utente e non introduce nuovi concetti, PARRY conduce la conversazione, mentre il suo interlocutore pone domande.

Descriviamo le principali regole che segue PARRY. Alcune parole o frasi vengono riscritte più volte cambiando solo la forma, non il significato. Le parole sconosciute, introdotte dall'interlocutore, vengono trascurate; per questo le risposte a volte non sono pertinenti: tuttavia in questi casi PARRY parla sempre degli stessi argomenti, come un vero paranoico.

Le risposte sono scelte in base all'input, quando esso viene compreso, e ai valori di alcune variabili di stato, dette "affect variables": i tre modi di comportamento sono "fear, anger, mistrust". A seconda di come si svolge la conversazione, queste variabili cambiano il loro valore, influenzando l'andamento della conversazione.

Input equivalenti, ma espressi in forma diversa, come ad esempio "What is your work?", "What do you do for a living?", "Do you have a job?" vengono trasformati dal parser nella stessa query: nell'esempio `(your work ?)`.

Alcune procedure cercano riferimenti ad aree sensitive, insinuazioni di malattia mentale e "flare concepts" (horses - horse-racing - gambling - bookies - underworld - Mafia), in modo da tornare sempre sugli stessi argomenti e cambiare il valore delle "affect variables".

3.5 SHRDLU

SHRDLU, come ELIZA e PARRY, è un programma per computer che ha come obbiettivo la comprensione del linguaggio naturale. La sua caratteristica principale, che lo differenzia da ELIZA, è un sistema di conoscenza dell'universo oggetto della conversazione.

È stato sviluppato da Terry Winograd nel 1968-70; il suo nome deriva dalla disposizione delle lettere sulla macchina Linotype.

L'utente può dialogare e dare istruzioni a SHRDLU di muovere vari oggetti nel piccolo mondo del programma, fatto di piramidi, cubi, sfere, ecc. e chiamato *blocks world*. Questo mondo e gli oggetti che contiene, come anche i movimenti compiuti dal programma, sono virtuali.

Winograd descrive così il suo programma: "Il sistema risponde a domande, esegue comandi, e accetta informazioni attraverso un dialogo interattivo in inglese... Il sistema contiene un parser, una grammatica per riconoscere l'inglese, dei programmi per analizzare la semantica, e un sistema per risolvere problemi... Esso può ricordare e discutere le sue azioni, oltre che compierle... La conoscenza

è rappresentata nella forma di procedure, invece che con tabelle di regole o liste di schemi".

SHRDLU si basa su quattro idee fondamentali. Innanzitutto il suo mondo è così semplice che per descriverlo bastano circa 50 parole, tra nomi (cubo, cono, ecc.), aggettivi (grande, verde, ecc.) e verbi (muovere, posare, ecc.). Le possibili combinazioni di queste parole sono abbastanza semplici e limitate, dunque è relativamente facile far fare al programma quello che si desidera.

SHRDLU è dotato di una memoria di base per completare le frasi senza contesto. Ad esempio, si potrebbe chiedergli "metti il cono verde sul parallelepipedo rosso" e, subito dopo, "rimuovi il cono": nella seconda frase "il cono" deve essere identificato come quello di cui si è appena parlato. Inoltre si possono fare domande al programma, relative alla storia, come "hai mosso qualcosa prima del cono?", a cui verrà risposto cercando nella memoria. SHRDLU ha una comprensione, almeno parziale di "it", che può riferirsi sia ad oggetti che ad azioni.

Un secondo risultato di questa memoria, insieme alle regole di cui è dotato, rendono possibile rispondere a domande relative a ciò che è possibile o impossibile fare: per esempio, SHRDLU deduce che è possibile impilare dei cubi, se ce ne sono due impilati, mentre fallisce se tenta di impilare due piramidi. Infatti il micro-mondo del programma contiene le regole di base della fisica, come la forza di gravità.

Il programma, infine, ricorda i nomi dati agli oggetti. Ad esempio, si potrebbe dirgli che "un campanile è una piccola piramide posata sulla cima di un alto parallelepipedo"; in seguito SHRDLU potrà rispondere a domande riguardanti i campanili e costruirne di nuovi.

Per vedere come queste quattro idee si applicano in pratica, di seguito è riportata una conversazione dimostrativa delle capacità di SHRDLU. Nella figura 5 è disegnata la situazione iniziale del mondo in cui il programma opera.

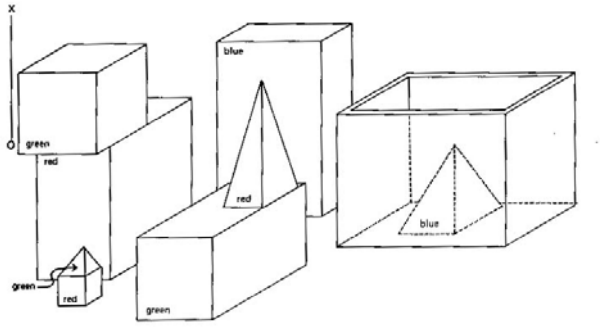


Figura 5: Situazione iniziale del mondo di SHRDLU

```
Person: PICK UP A BIG RED BLOCK.
Computer: OK. (does it)
Person: GRASP THE PYRAMID.
Computer: I DON'T UNDERSTAND WHICH PYRAMID YOU MEAN.
```

SHRDLU obbedisce ai comandi, se sono chiari; risponde “OK” quando esegue un comando. Per spostare dei blocchi a volte è necessario spostare prima i blocchi posati su di essi: il sistema se ne accorge e lo fa: per esempio per obbedire al primo comando ha dovuto prima spostare il solido verde (figura 5). Quando il comando non è chiaro, il programma non lo esegue: ad esempio non sposta la piramide perchè ce ne sono tre e l’utente non specifica quale vuole spostare.

```
Person (changing their mind): 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)
```

Il sistema deve decidere cosa si intende per “one” e per “it”; inoltre deve dedurre che sta tenendo il blocco che non ha posato in nessun posto. Nella figura 6 si vede come è cambiata la situazione.

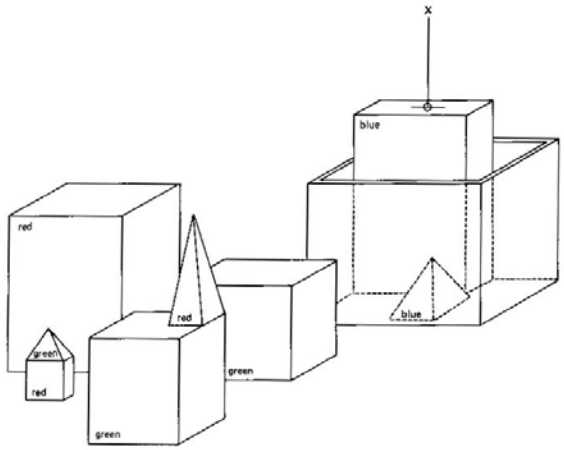


Figura 6: I blocchi vengono spostati

```
Person: WHAT DOES THE BOX CONTAIN?
Computer: THE BLUE PYRAMID AND THE BLUE BLOCK.
Person: WHAT IS THE PYRAMID SUPPORTED BY?
Computer: THE BOX.
```

Il sistema risponde a domande riguardanti la situazione del micro-mondo.

In questo caso capisce a quale piramide ci si riferisce, anche senza che venga specificata: assume che sia quella di cui si sta parlando.

```
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.
```

La frase "at least one of them" deve essere analizzata: si riferisce alla domanda precedente. I nomi degli oggetti possono essere complicati da numeri: ad esempio "at least one", "more than three", ecc.

La memoria del passato è usata per capire quale solido sia "the one which I told you to pick up".

Il passivo può essere usato senza specificare il complemento d'agente.

```
Person: CAN THE TABLE PICK UP BLOCKS?
Computer: NO.
Person: CAN A PYRAMID BE SUPPORTED BY A BLOCK?
Computer: YES.
```

Il sistema risponde a domande su cosa è possibile fare e cosa no. Se le regole semantiche rendono senza senso una domanda, la risposta ad essa deve essere "no".

C'è un esempio di piramide appoggiata su un parallelepipedo, dunque la risposta alla seconda domanda deve essere positiva.

```
Person: CAN A PYRAMID SUPPORT A PYRAMID?
Computer: I DON'T KNOW.
Person: STACK UP TWO PYRAMIDS.
Computer: I CAN'T.
```

Ci sono domande, su cosa sia possibile fare, a cui non sa rispondere. Allora si può provare a fargli compiere l'azione richiesta, per vedere se sia possibile.

```
Person: WILL YOU PLEASE STACK UP BOTH OF THE RED BLOCKS
 AND EITHER A GREEN CUBE OR A PYRAMID?
Computer: OK. (does it)
```

I comandi vengono eseguiti anche se sono posti in forma di domanda.

```
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.
```

Parole come "littlest" non sono presenti nel dizionario, ma vengono interpretate grazie alla radice "little".

Il sistema interpreta frasi complesse; alcune parole, come "support" possono essere usate senza confusione, anche se sono sia verbi che nomi.

```
Person: WHAT COLOR IS IT?
Computer: BY "IT", I ASSUME YOU MEAN THE SHORTEST THING
 THE TALLEST PYRAMID'S SUPPORT SUPPORTS.
Computer: RED
```

In questo caso interpreta non interpreta "it" come la "green pyramid", che è l'ultimo oggetto di cui si è parlato, perchè non ha senso chiedere il colore di un oggetto che è stato appena identificato grazie a quel colore.

```
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 - ANWHERE ON TOP OF ?

Person: 2
Computer: THREE OF THEM.
```

Se ci sono diverse interpretazioni possibili e nessuna è migliore delle altre, il sistema chiede spiegazioni all'utente. Le parafrasi del significato di "on" sono inserite nel dizionario. Altri casi ambigui possono darsi con l'uso di "it" o "they".

```
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.
```

Il sistema risponde a domande riguardanti quello che ha fatto in precedenza: riesce a capire a quale evento si fa riferimento.

Per rispondere a domande come "Why?", il programma ricorda le frasi passate.

"That" nell'ultima frase si riferisce ad un'azione, non ad un oggetto.

```
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)
```

Questa frase potrebbe essere ambigua, ma poichè l'interpretazione "put the blue pyramid which is on the block in the box" non ha senso nello scenario corrente, viene scelta l'altra interpretazione possibile: "put the blue pyramid on the block which is on the box".

```
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.
```

Strutture come "bigger than", "as wide as" e quantificatori come "any" o "some" vengono capiti.

```
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)
```

Il sistema si interrompe e chiede spiegazioni se trova una parola che non riesce ad interpretare. Si possono definire parole nuove, che vengono memorizzate; il sistema ora usa il nuovo vocabolo e costruisce le strutture di cui ha imparato il nome.

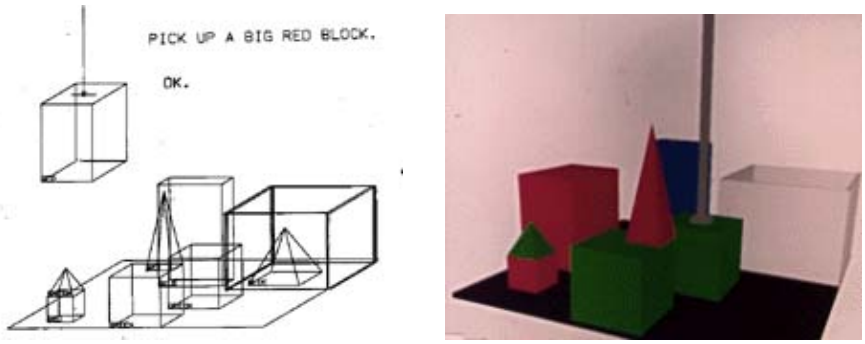
Si possono dare nomi agli oggetti.

```
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!
```

La dimostrazione delle capacità di SHRDLU fornì una prova delle potenzialità dell'intelligenza artificiale: essendo uno dei primi programmi del genere, esso portò ad un eccessivo ottimismo da parte dei ricercatori, che tuttavia si spense presto, quando con altri sistemi cercarono di trattare situazioni più realistiche e vicine alla realtà, anche ambigue e complesse.

Nella figura 7 vediamo l'interfaccia originale del programma e quella disegnata dai ricercatori dell'università dello Utah, con il rendering in 3D.



(a) Display originale (b) Dopo il rendering

Figura 7: Come appare SHRDLU

4 Algoritmi minimax

L'intelligenza artificiale viene impiegata anche nei giochi dove uno o più giocatori sono gestiti da un computer. Analizziamo i giochi deterministici con informazione perfetta, cioè quelli in cui si conoscono all'inizio e si possono determinare tutte le possibili mosse; inoltre lo svolgimento del gioco non dipende da fattori casuali (carte, dadi, ecc.). Ci limitiamo a due giocatori, che alternativamente compiono una mossa: giochi di questo tipo sono il tris, gli scacchi, il go, ecc.

Possiamo formulare il problema di questi giochi come un problema di ricerca: la situazione di gioco corrisponde alla situazione della ricerca; le mosse possibili corrispondono agli operatori; gli stati finali sono quelli in cui il gioco è vinto, perso o pareggiato. L'obiettivo è trovare una strategia, cioè una sequenza di mosse che ci faccia vincere la partita.

Dobbiamo ricordare che il nostro avversario cerca di compiere mosse favorevoli a lui e sfavorevoli a noi; dobbiamo simulare le sue decisioni. Dunque noi saremo il giocatore max, che cerca di massimizzare il proprio vantaggio, e l'avversario sarà il giocatore min, perchè tenterà di minimizzarlo.

Il computer, per compiere la sua mossa, dovrebbe in teoria analizzare tutte le possibili alternative e scegliere la migliore. Lo svolgimento del gioco si schematizza in un albero. Ogni nodo rappresenta una situazione di gioco, ogni arco che collega due nodi è una possibile mossa. Le foglie sono situazioni finali, in cui uno dei due giocatori vince.

Una soluzione ottima del problema significa compiere delle mosse ed arrivare ad una vittoria certa; per ottenerla è necessario ispezionare tutto l'albero del gioco. In un gioco come gli scacchi, però, l'albero completo ha un numero di nodi dell'ordine di 10^{40}: nessun computer potrà mai analizzarlo tutto in un tempo utile; avrebbe infatti bisogno di molti secoli. Dunque la mossa del computer dovrà basarsi solo su una parte dell'albero: poichè è impossibile ottenere soluzioni ottime, l'obiettivo si sposta su decisioni "buone", calcolate in un tempo accettabile.

Viene generata ed analizzata solo una parte del sottoalbero radicato nel nodo corrispondente alla situazione corrente: ogni foglia del sottoalbero ha un valore (euristico) che indica quanto è favorevole per ogni giocatore la situazione descritta da quel nodo.

La funzione di valutazione misura quanto è desiderabile una data situazione di gioco, ovvero quante sono le possibilità di vittoria a partire da quella situazione. Ad esempio, nel gioco del tris si può usare come funzione di valutazione la differenza tra il numero dei percorsi vincenti che possono essere ottenuti dalla situazione corrente e il numero di percorsi che portano alla sconfitta.

La situazione della figura 8 ha un valore pari a 4-2=2; infatti si può vincere in quattro modi (due linee orizzontali, una verticale e una diagonale), mentre l'avversario ha solo due possibilità di vittoria (una linea orizzontale e una verticale.

L'algoritmo minimax si può descrivere in questo modo:

1. valutando n livelli successivi alla situazione attuale, assegna dei valori alle foglie, in base all'euristica applicata

2. assegna il valore a ciascun nodo padre:

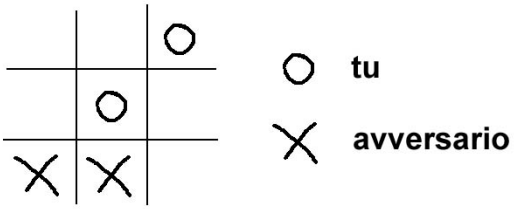


Figura 8: Situazione di gioco

- se il padre è un nodo max, assegnagli il massimo tra i valori dei figli
- se il padre è un nodo min, assegnagli il minimo tra i valori dei figli

3. per il nodo radice (la situazione attuale), la mossa migliore è quella del figlio con il valore massimo.

La figura 9 illustra un albero minimax.

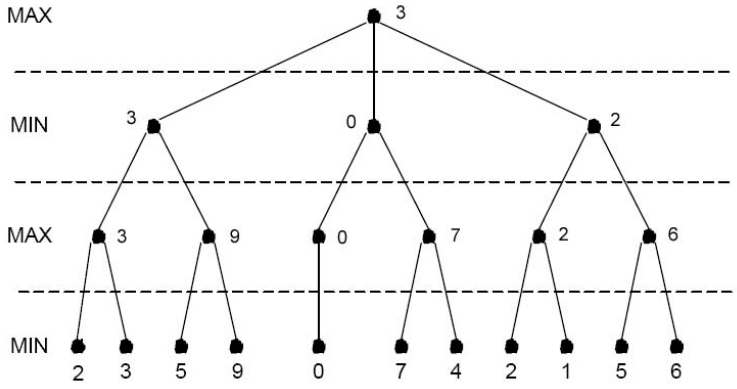


Figura 9: Algoritmo minimax

Vediamo ora come si applica l'algoritmo minimax al gioco del tris. La figura 10 rappresenta l'albero per la mossa iniziale di questo gioco: i casi simmetrici sono considerati uguali.

La mossa iniziale è fatta dal giocatore A, che sceglie la mossa del massimo vantaggio: egli sa che l'altro giocatore B sceglierà quella che dà ad A il minimo vantaggio. Dall'albero illustrato in figura 10 si può ricavare la seguente tabella:

Mossa di A	Possibili mosse di B	Mossa di B
-1	1, 0, -1	-1
-2	-1, 0, -2	-2
1	1, 2	1

A deve scegliere la mossa in modo che la mossa di B generi la situazione più favorevole possibile ad A: dunque A sceglierà la mossa 1.

Le figure 11 e 12 illustrano gli alberi per le mosse seguenti.

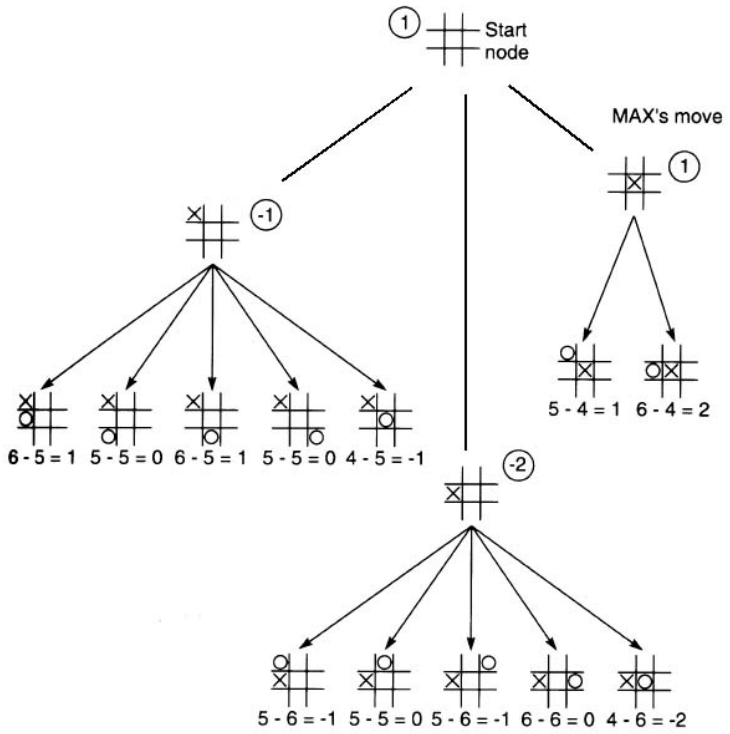


Figura 10: Albero di gioco: mossa iniziale

In questo modo abbiamo scoperto che nel tris il giocatore che inizia ha una strategia vincente, ovvero riesce a vincere seguendo le mosse descritte negli alberi visti.

Se l'albero di gioco è finito, l'algoritmo minimax è completo, cioè trova una strategia da applicare. Inoltre esso è ottimo, se l'avversario è ottimo, ovvero se egli si comporta nel modo che abbiamo supposto, cercando di minimizzare il nostro vantaggio.

4.1 α-β pruning

Un miglioramento di questo algoritmo è il "$\alpha - \beta$ pruning": invece di valutare tutti i casi, ci si ferma prima se...

Questa procedura non cambia il risultato finale, ed ha il vantaggio di essere più veloce se l'ordine con cui vengono esaminate le mosse è "buono"; la complessità temporale può dimezzare se l'ordinamento è perfetto.

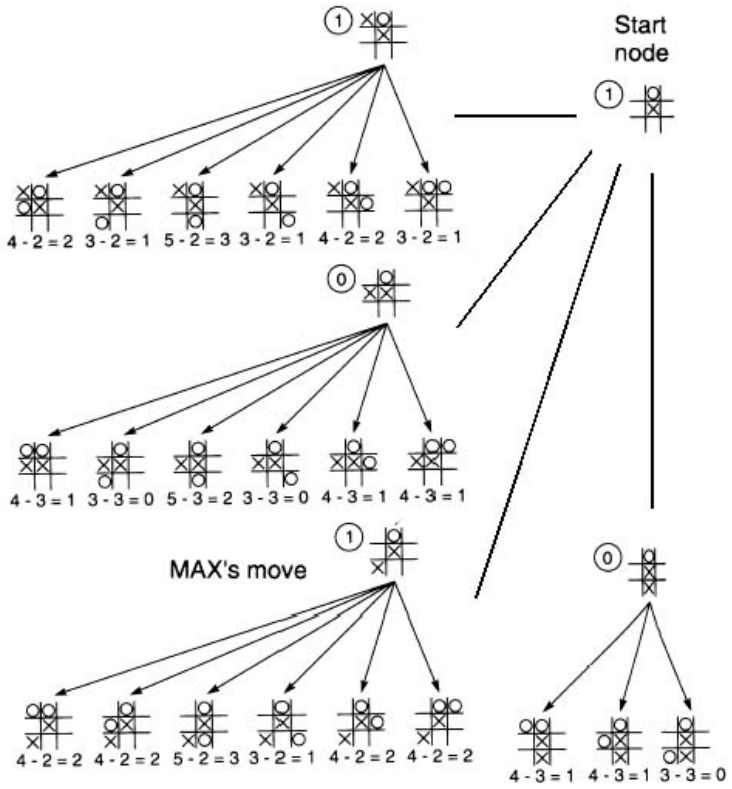


Figura 11: Albero di gioco: seconda mossa

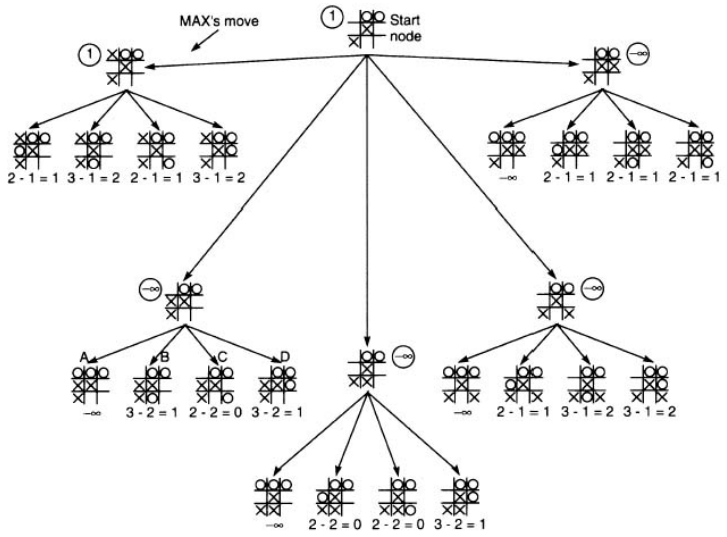


Figura 12: Albero di gioco: ultima mossa

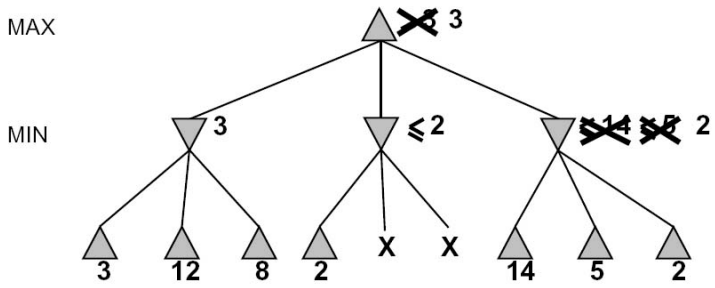


Figura 13: Strategia $\alpha - \beta$

Riferimenti bibliografici

[1] *http://en.wikipedia.org/wiki/*
Enciclopedia on-line.

[2] *http://www.cs.man.ac.uk/ schalk/3192/*
Pagina del corso di teoria dei giochi dell'università di Manchester.

[3] *http://www.loebner.net/Prizef/TuringArticle.html*
L'articolo dove Turing descrive il suo test.

[4] *http://members.aol.com/NeoNoetics/MindsBrainsPrograms.html*
L'articolo di Searle sulla stanza cinese.

[5] *http://www.psych.utoronto.ca/ reingold/courses/ai/turing.html*
Test di Turing.

[6] *http://www.Planet-Source-Code.com/vb/scripts/ShowCode.asp?txtCodeId =5369&lngWId=3*
Una semplice implementazione di ELIZA, con codice e spiegazione.

[7] *http://www-ai.ijs.si/eliza/eliza.html*
Pagina dove si può parlare con ELIZA.

[8] *http://www.manifestation.com/neurotoys/eliza.php3*
Pagina dove si può parlare con ELIZA.

[9] *http://www.cs.queensu.ca/home/cisc352/weizenbaum.pdf*
L'articolo di Weizenbaum che descrive ELIZA.

[10] *http://www.ee.vt.edu/ ee4524hv/*
Pagina del corso di Artificial intelligence and pattern recognition dell'università di .....

[11] *http://http.cs.berkeley.edu/ russell/ai.html*
Links a pagine che trattano di AI.

[12] *http://hci.stanford.edu/ winograd/shrdlu/*
SHRDLU: esempio di conversazione e download del codice.

[13] *http://www.semaphorecorp.com/misc/shrdlu.html*
Pagina su SHRDLU; persone che hanno lavorato a questo programma.