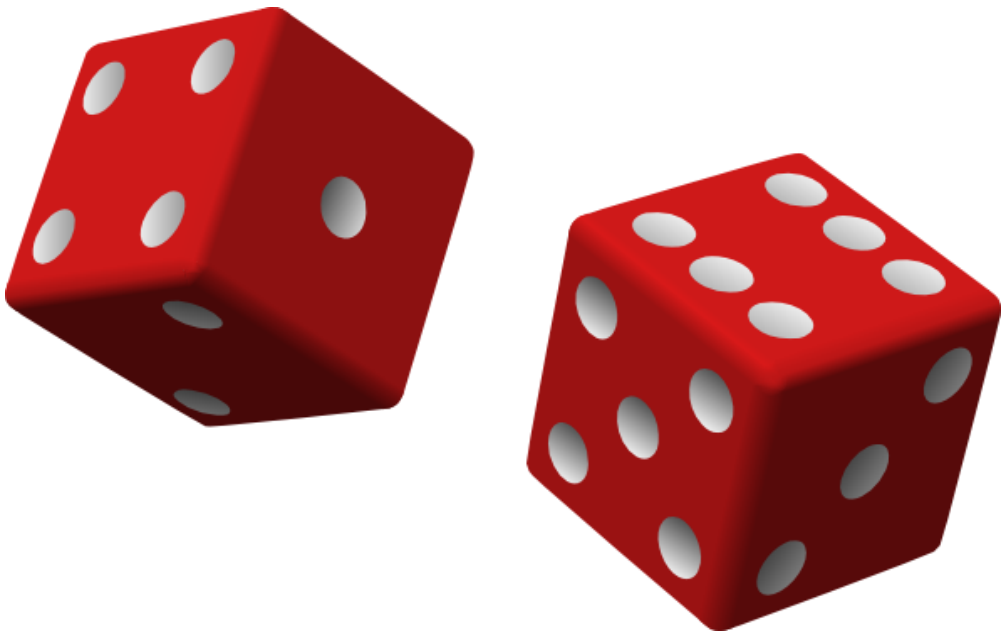
I.I.S. “Benedetto Castelli” - Brescia
Tesina per l’Esame di Stato - a.s. 2015/2016
Classe 5^ sezione AI

Pietro Venturini

IL CASO



“Ancora oggi molte persone d’ingegno non riescono ad accettare e neppure a comprendere come la selezione, da sola, abbia potuto trarre da una fonte di rumore tutte le musiche della biosfera” (Jacques Monod, 2001).

INDICE

INFORMATICA

SISTEMI E RETI

LETTERATURA

KANJI APP

Premessa

La scelta dell'argomento di questa tesina nasce dall'interesse che il sistema di generazione di numeri casuali in informatica ha suscitato in me quasi un anno fa durante un approfondimento personale dell'argomento.
La costruzione del progetto, una *web application*, è iniziata precedentemente alla stesura della tesina scritta e, inizialmente, il suo unico scopo era quello di ricercare alcuni caratteri giapponesi chiamati *kanji* partendo dal loro significato.

La scelta del progetto legato ad una lingua straniera, invece, è nata dalla mia passione per i viaggi e per le altre culture, ho voluto sfruttare l'occasione di utilizzare tecnologie e linguaggi di programmazione a me nuovi per imparare, oltre a questi, una nuova lingua.

Per poter agganciare il progetto all'argomento della tesina, si è resa necessaria l'implementazione di una funzionalità aggiuntiva al programma: la generazione di testo casuale tramite l'algoritmo delle catene di Markov. Si è trattato di un tema per nulla semplice che ha richiesto mesi di lavoro e ricerche, soprattutto per la sua implementazione all'interno del progetto. Con l'aiuto di alcuni professori è stato possibile effettuare collegamenti interdisciplinari inserendo diverse discipline, non solo tecniche, all'interno della tesina.

INTRODUZIONE

Fino al XVII secolo ogni fenomeno era interpretato sulla base dei principi religiosi e la casualità degli eventi era associata al mistero del disegno divino. Si credeva che il destino fosse già scritto, e che fosse inevitabile sfuggirci.

Con l'illuminismo e l'affermazione della scienza moderna cambia sia la visione della realtà e sia quella del caso. Il caso non è più il mistero divino ma soltanto l'ignoranza dell'uomo sulle leggi che regolano l'universo.
Nella visione scientifica si afferma il determinismo: ogni trasformazione può essere interpretata con assoluta certezza tramite leggi semplici e universali, spetta comunque all'uomo il compito di indagare e scoprire le leggi e i nessi tra gli eventi.
Il fisico Albert Einstein si pronuncia a riguardo con la famosa frase *“Dio non gioca a dadi”.*
Il caso diventa così il sinonimo dell'ignoranza, ricorrendo al calcolo probabilistico soltanto in rari casi.

Nella prima metà del Novecento le nuove scoperte scientifiche delineano un mondo controllato da leggi complesse. Il caso non è più soltanto l'ignoranza dell'uomo ma, molte volte, è la natura stessa della realtà. Ad esempio, non si può prevedere con esattezza le condizioni meteorologiche di un luogo a lungo termine. Per questo, nel corso del Novecento Einstein si accorge che *"Dio gioca a dadi"*. Il caso e la casualità entrano così a far parte del linguaggio scientifico e della stessa scienza.

MATEMATICA

I campi di applicazione della scienza del caso, detta anche teoria matematica della probabilità, sono svariati e comprendono, ad esempio, le scommesse, la definizione di rischio nucleare, i sondaggi, le previsioni meteorologiche o economiche…
Il fine di questa disciplina è prevedere il verificarsi o meno di un evento servendosi del calcolo probabilistico.
In probabilità si considera un fenomeno osservabile esclusivamente dal punto di vista della possibilità o meno del suo verificarsi, prescindendo dalla sua natura. Tra due estremi, detti **evento certo** (ad esempio: lanciando un dado a sei facce si ottiene un numero compreso tra 1 e 6) ed **evento impossibile** (ottenere 1 come somma dal lancio di due dadi), si collocano eventi più o meno probabili detti **aleatori** (casuali).
Dopo aver introdotto alcuni aspetti teorici riguardanti la probabilità, verrà presentata una sua applicazione nel gioco d'azzardo e la teoria dei processi Markoviani.

La probabilità

Definizione generale

La probabilità è un numero utilizzato per quantificare la facilità o la difficoltà con cui un evento può verificarsi.
Può essere calcolata come **il rapporto tra il numero di casi favorevoli e il numero di casi possibili.**

$$P = \frac{casi\ favorevoli}{casi\ possibili}$$

L'insieme di tutti i possibili esiti di un evento è chiamato **spazio campionario** e viene indicato con il simbolo Ω.
Ogni sottoinsieme di Ω è chiamato **evento** e può essere:

- **Certo:** quando coincide con lo spazio campionario.
- **Elementare:** quando è costituito da un solo elemento dello spazio campionario.
- **Impossibile:** quando coincide con l'insieme vuoto.

Ulteriori definizioni sono date da:

- **Evento Unione:** dati due eventi A e B, si indica con $A \cup B$ l'evento che si realizza quando si realizzano gli eventi A o B (o entrambi).
- **Evento Intersezione:** dati due eventi A e B, si indica con $A \cap B$ l'evento che si realizza quando si realizzano simultaneamente gli eventi A e B.
- **Eventi incompatibili:** vengono così chiamati due eventi la cui intersezione è l'evento impossibile (gli eventi non possono verificarsi contemporaneamente).

- **Evento contrario:** si definisce evento contrario di A, e si indica con A, l'evento che si realizza quando **non** si realizza A, ossia l'evento rappresentato dal complementare di A.

Assiomi di probabilità

Un assioma è un principio, posto a fondamento di una teoria deduttiva, che è talmente ovvio da non aver bisogno di essere dimostrato.
La probabilità $P(E)$ di un evento E è un numero reale che verifica i seguenti assiomi:

$$0 \leq P(E) \leq 1$$

Se Ω è l'evento certo, allora $P(\Omega) = 1$.

Se gli eventi A e B sono incompatibili (non possono verificarsi contemporaneamente) allora:

$$P(A \cap B) = P(A) * P(B)$$

Teoremi sulla probabilità

I teoremi riguardanti eventi indipendenti (ossia che il verificarsi di uno non influisce sul verificarsi o meno dell'altro) sono:

- **Intersezione di due eventi**

  $P(A \cap B) = P(A) * P(B)$

- **Unione di due eventi**

  $P(A \cup B) = P(A) + P(B) - P(A \cap B)$

Probabilità condizionata

Essa interviene quando si calcola $p(A)$ sapendo che si è verificato un evento B che influisce sulla probabilità che si verifichi A.
Si indica con $p(A|B)$ e si calcola con la formula:

$$p(A|B) = \frac{p(A \cap B)}{p(B)}$$

Da cui la formula delle **probabilità composte:**

$$p(A \cap B) = p(B) \cdot p(A|B)$$

Il calcolo combinatorio

Con calcolo combinatorio si intende la branca della matematica che si occupa di studiare i modi per raggruppare e ordinare degli insiemi finiti di oggetti. Esso viene utilizzato solitamente al fine di rispondere a tutte quelle domande del tipo “In quanti modi...”, “Quanti sono...”, “Quante possibili combinazioni...”, eccetera.

Prima di affrontare qualsiasi problema di calcolo combinatorio è necessario porsi le seguenti domande:

1. L’ordinamento è importante? (Es. la configurazione {x, y, z} è uguale a {y, z, x}?)
2. È possibile utilizzare lo stesso oggetto per più configurazioni?

In base al tipo di problema in questione, è possibile utilizzare più tecniche di raggruppamento tra cui:

- Permutazioni (con e senza ripetizione di oggetti)
- Disposizioni (con e senza ripetizione di oggetti)
- Combinazioni (con e senza ripetizione di oggetti)

Permutazioni

Per definizione, dato un insieme N di n elementi distinti, una **permutazione semplice** è una sequenza ordinata di n oggetti nella quale ogni elemento viene preso una ed una sola volta. Ciascuna delle possibili permutazioni ottenibili da un insieme di partenza conterrà tutti e soli gli elementi n dati e differirà da ogni altro raggruppamento per l’ordine secondo il quale i propri elementi sono disposti.

- Numero di oggetti = Numero di posti
- Conta l’ordine

$$P_n = n \cdot (n-1) \cdot (n-2) \cdot \ldots \cdot 1 \; = \; n!$$

Se l’insieme di partenza N contiene elementi che si ripetono, si otterranno delle sequenze duplicate. Per ovviare a questo problema è necessario tener conto del numero di volte $k_1, k_2, \ldots, k_3$ che ciascun oggetto si ripete utilizzando la formula delle **permutazioni con ripetizione di oggetti**.

$$P_n^{k_1,k2,\ldots,k_r} = \frac{n!}{k_1!k_2! \cdots k_r!}$$

Disposizioni

Dato un insieme N di n elementi distinti, una **disposizione semplice** è una sequenza ordinata di k oggetti con $k \leq n$ nella quale ogni elemento viene preso una ed una sola volta.

Ciascuna delle possibili permutazioni ottenibili da un insieme di partenza conterrà tutti e soli gli elementi n dati e differirà da ogni altro raggruppamento per l'ordine secondo il quale i propri elementi sono disposti.

- $Numero\ di\ oggetti \neq Numero\ di\ posti$
- Conta l'ordine

$$D_{n,k} = \frac{n!}{(n-k)!} = n \cdot (n-1) \cdot .... \cdot (n-k+1)$$

Se si volessero ottenere delle ripetizioni di uno stesso elemento appartenente all'insieme di partenza N è necessario la formula delle **disposizioni con ripetizione di oggetti**.

$$D^r_{n,k} = n^k$$

Combinazioni

Dato un insieme N di n elementi distinti, sono chiamate **combinazioni semplici** tutti i raggruppamenti che si possono formare con $k \leq n$ dei suoi elementi in modo che:

- Tutti i k elementi di ogni raggruppamento siano distinti tra loro
- Ogni raggruppamento sia diverso dagli altri per almeno un elemento e non per via del diverso ordine in cui sono disposti.

Esse vengono indicate con la sigla $C_{(n,k)}$ e hanno le seguenti caratteristiche:

- $Numero\ di\ oggetti \neq Numero\ di\ posti$
- Non conta l'ordine

$$C_{n,k} = \frac{n!}{k!(n-k)!}$$

Quando l'ordine non è importante ma è possibile avere componenti ripetute si parla di **combinazioni con ripetizione**.

$$C^r_{n,k} = \frac{(n+k-1)!}{k!(n-1)!}$$

La probabilità nel gioco d'azzardo

Chiunque abbia incontrato persone che giocano si sarà reso conto, almeno una volta, che queste persone tendono ad avere **convinzioni errate** che portano loro ad avere atteggiamenti deleteri che possono comportare ingenti **perdite di denaro** e, nei casi peggiori, una **dipendenza** accanita dal gioco stesso.
L'obiettivo degli esempi trattati di seguito, proposti nel corso *BetOnMath*[1] organizzato dal Politecnico di Milano, è descrivere diversi aspetti legati al gioco d'azzardo quali la probabilità di vincita, il comportamento del giocatore di fronte alle "quasi vincite" e i guadagni derivanti da questi giochi.

Esempi di eventi

Come è stato spiegato nel capitolo precedente, la probabilità circa il verificarsi o meno di un evento può essere calcolata come **il rapporto tra il numero di casi favorevoli e il numero di casi possibili.**
Ad esempio la probabilità di ottenere 5 lanciando un dado a 6 facce e' ⅙ perché una sola è la faccia con il 5, il caso favorevole e 6 sono le facce possibili.
Tra gli esempi trattati vi sono la probabilità di incontrare una persona nata il proprio stesso giorno dell'anno (1), parcheggiare vicino ad una specifica persona in un parcheggio da 5000 posti (2), indovinare il pin del bancomat digitando una sequenza a caso (3), e far cadere una moneta due volte consecutive di taglio (4).

1. Approssimando e quindi escludendo gli anni bisestili, i casi possibili sono 365 ossia i giorni di un anno. Il caso favore è solo uno quindi la probabilità è $\frac{1}{365}$.

2. Nel parcheggio ci sono 4999 posti possibili, i casi favorevoli sono 2 perché vi è la possibilità di parcheggiare sia alla sua sinistra che alla sua destra.
   La probabilità è circa $\frac{1}{2500}$.

3. Sulla tastiera del bancomat ci sono 10 cifre da 0 a 9. I casi possibili sono 10^5 perché vi sono 10 possibilità per ognuna delle cifre da digitare.
   Solo una sequenza è quella corretta quindi la probabilità di azzeccare il pin è $\frac{1}{100.000}$.

4. La probabilità che una moneta cada di taglio è $\frac{1}{6.000}$ quindi la probabilità che la moneta cada due volte di taglio è 1 su 6.000 al quadrato, cioè 1 su $\frac{1}{36.000.000}$

Vincere al gioco d'azzardo

Per legge, lo Stato italiano ha l'obbligo di dichiarare le probabilità di vincita dei giochi. Ad esempio. la probabilità dichiarata di trovare un premio da 5€ sui biglietti del Gratti e Vinci 'Il Miliardario' (costo del biglietto: 5€) è del 28,7%, ovvero circa ¼ .
Tuttavia **trovare un premio dello stesso valore del grattino non è una vincita**!
La probabilità di vincere un premio superiore a 1000€ è circa $\frac{1}{5.600}$ ossia circa 15 volte meno probabile di incontrare una persona che festeggi il compleanno il proprio stesso giorno, e addirittura due volte meno probabile di parcheggiare di fianco ad una persona specifica in un parcheggio da 5000 posti.
Un altro gioco popolare è il VinciCasa, nel quale si vince indovinando una sequenza di 5 numeri tra un insieme di 40.
Il numero delle diverse sequenze di 5 numeri che si possono formare da un insieme di 40 è calcolabile utilizzando il metodo delle disposizioni di 40 elementi di classe 5, ossia circa 658.000, quindi la probabilità di vincita è $\frac{1}{658.000}$, ossia 1800 volte meno probabile di incontrare una persona nata nel proprio stesso giorno e 6 volte meno probabile di azzeccare un PIN del bancomat.
Infine nel SuperEnalotto fare 6 significa indovinare una sequenza di 6 numeri da un insieme di 90 numeri: un evento con 622.614.630 possibilità differenti, quindi la probabilità è pari a $\frac{1}{622.614.630}$.

Per capire meglio il significato di questi numeri si possono utilizzare mezzi come **il tempo**[1], **le carte da gioco**[2] e **i gruppi di persone**[3].

1. Facendo un tentativo a settimana di vincere un premio superiore a 1000€ al Gratta Vinci, come già detto, con una probabilità di vincita pari a $\frac{1}{5600}$, dopo 5600 settimane, ossia **104 anni**, ci si sarà aggiudicati il premio, in media, una sola volta.
   Giocando a VinciCasa con una sola probabilità su 658.000 di vincere una casa, dovrebbero passare **12.653 anni per vincere in media una sola volta**.
   Per giocare tutte le 622.614.630 sequenze possibili al SuperEnalotto, giocando sempre una volta a settimana, **passerebbero quasi 12 milioni di anni prima di vincere una sola volta in media il Jackpot**!

2. Immaginando che il totale dei casi possibili associato ad una certa probabilità corrisponda ad un determinato numero di carte, il mazzo corrispondente alle possibilità di vincita superiori a 1000€ al Gratta e Vinci sarebbe formato da 5600

carte che impilate una sull'altra formerebbero un mazzo di carte alto circa 2 metri nel quale solo una, in media, è la carta vincente.
Invece al VinciCasa i circa 658.000 eventi possibili formerebbero un mazzo di carte alto circa 247 metri, quasi come la Tour Eiffel, alta 300 metri.
Nel SuperEnalotto i 622.614.630 eventi possibili corrisponderebbero addirittura ad un **mazzo di carte alto 230.000 metri**, più della metà della distanza tra la Terra e la Luna.

3. Quantificare la probabilità di vincita in relazione al numero delle persone che dovrebbero giocare affinché in media una sola di loro vinca, significa che tutti i 5.600 abitanti di un piccolo comune dovrebbero giocare un Gratta e Vinci ciascuno per far sì che, in media, solo uno di loro si aggiudichi un premio superiore a 1000€.
   Per ottenere il premio massimo al Vinci casa sarebbero necessari in media 658.000 tentativi per vincere in media una sola casa, ovvero tutti gli abitanti della città di Palermo.
   Per vincere infine al SuperEnalotto servirebbero in media 622.614.630 tentativi, ossia tutti i cittadini europei dovrebbero giocare contemporaneamente e in media solo uno vincerebbe il Jackpot.

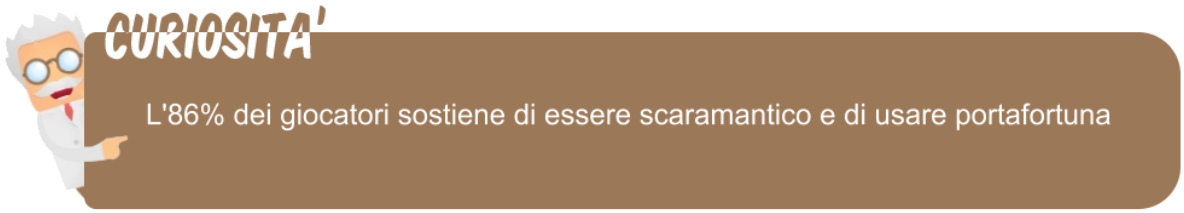
Il comportamento dei giocatori e la quasi vincita

Quando giocando il giocatore si avvicina alla vincita, credendo di aver quasi vinto, corre il rischio di continuare a giocare con la convinzione che il prossimo tentativo sarà quello vincente.
In realtà la quasi vincita non esiste, o si vince o si perde.

Giocando al SuperEnalotto, una sequenza come 11,26,41,59,65,88 ha la **stessa** identica **probabilità** della sequenza 1,2,3,4,5.
Siamo portati a credere il contrario perché alcune sequenze sembrano troppo regolari per essere frutto di un estrazione casuale.
Quindi ci sembra che alcune sequenze siano più rappresentative di altre e siamo portati a pensare che abbiano una probabilità maggiore di uscire.
La conseguenza di affidarci a questa conoscenza errata è che tendiamo a sovrastimare in modo erroneo la probabilità di alcuni eventi sui quali, proprio per questo motivo, siamo portati a scommettere, con l'illusione di poter vincere più facilmente.

La fallacia del giocatore

La conoscenza che porta a scommettere sui numeri ritardatari che non escono da molto tempo o sul fatto che alla roulette sia uscito per troppe volte il nero e si pensa che al prossimo turno esca il rosso, è errata ma ha una grande forza e prende il nome di fallacia del giocatore.



La Legge dei Grandi Numeri

Nel lancio di una moneta, dove le possibilità sono testa o croce, è molto comune credere che lo scenario più verosimile sia quello in cui la differenza tra teste e croci tenda ad annullarsi all'aumentare dei lanci
Questo errore è un tipico esempio di "fallacia del giocatore" ed è figlio di una interpretazione non corretta di quella che è nota come la Legge dei Grandi Numeri.

Infatti la Legge dei Grandi Numeri stabilisce che per un numero di lanci sempre più grande, **la frequenza di uscita di teste deve tendere alla sua probabilità teorica**:

$$\frac{Numero\ di\ Teste}{Numero\ di\ Lanci} \approx 50\%$$

Se dopo 20 lanci è uscito sempre testa, si avrà una differenza fra teste e croci di 20, e una frequenza di testa del 100%.
Se a 1000 lanci la differenza non e' scesa, per esempio è rimasta ancora 20, vuol dire che sono uscite 510 teste e 490 croci ovvero la frequenza di testa è scesa al 51%, molto vicina alla probabilità teorica del 50%.
Il giocatore si aspetta che la differenza si azzeri, tuttavia è la frequenza che tende a diventare stabile all'aumentare dei tentativi.
La differenza invece può assumere comportamenti alquanto imprevedibili.
La giocata successiva è comunque sempre imprevedibile!

Il costo del gioco: il concetto di iniquità

Come tutti i giochi d'azzardo, il Gratta e Vinci è iniquo, ossia è un gioco che restituisce ai giocatori solo una parte del denaro speso (in questo caso questo indice è pari al 70%).
Ad esempio, acquistando un Gratta e Vinci al mese nella speranza di conquistare un premio elevato, in 5 anni qualche volta si avrà trovato un biglietto vincente, ma la maggior parte delle volte non si avrà vinto nulla, spendendo 300€ ossia 60€ all'anno.

Comprando invece un Gratta e Vinci al giorno, dopo 10 giocate per un totale di 50€ spesi se ne avranno recuperati, in media, circa 35€.
Dopo un anno la spesa è salita a 1.800€, con un rientro in media di 1.260€ e di conseguenza una perdita di 540€.
Continuando a giocare si entra in un trend negativo che porta in media a perdere sempre più denaro con sempre minore probabilità di recuperarlo.
Per comprare 300 biglietti del Gratta e Vinci a settimana, si sono spesi 6.000€ al mese, tutti i mesi, per 5 anni: in totale si sono spesi 360.000€ ottenendo in media solo 252.000€ di premi, perdendone così ben 108.000€.

Per calcolare l'indice di equità si deve innanzitutto calcolare il premio medio, ovvero la media pesata dei premi associati a ciascun possibile esito del gioco.
Nel testa e croce gli esiti possibili sono solo due ed entrambi hanno il 50% di probabilità di verificarsi.
Per esempio: se esce testa si vincono 1,50 € e la probabilità è ½.
Se esce croce non si vince nulla, ovvero il premio è 0 €. La probabilità è sempre ½.
Il premio medio si calcola facendo la media dei premi pesata con la loro probabilità

$$1,5€ * 1/2 + 0€ * 1/2 = 0,75€$$

L'indice di equità è definito come $\frac{Premio\ Medio}{Costo\ della\ Giocata}$.

Se fossero uguali l'indice di equità sarebbe pari a 1, o 100%, e il gioco sarebbe equo.
Nell'esempio del testa e croce, dove il costo della giocata è pari a 1€, il gioco ha indice di equità pari a 0.75. Essendo inferiore a 1, il gioco è iniquo.
In poche parole si può dire che questo gioco restituisce in media il 75% dei soldi spesi per giocare.

Il problema dei Grandi Premi

I premi molto grandi confondono il giocatore. Sembra che basti una cifra alta per compensare una probabilità molto piccola.
Se per esempio si comprassero tutti i 30 milioni di biglietti di una emissione del Gratta e Vinci si spenderebbero in tutto 150 milioni di euro vincendo così tutto il montepremi ossia 105 milioni di euro.
La perdita è quindi di 45 milioni di euro ossia il 30%.
Il gioco è infatti studiato e costruito affinché il suo indice di equità sia del 70%.

CURIOSITA'

Nel 2012, solo in Italia, sono stati giocati più 86.000.000.000 di Euro nel gioco d'azzardo, e persi più di 24.000.000.000 €

Le catene di Markov

La teoria delle catene di Markov[2][3] è stata scoperta o introdotta, a seconda dei punti di vista, da un matematico russo, Andrej Andreevič Markov, nel 1906.
Una catena di Markov è un processo di tipo **stocastico** che gode della proprietà dei cosìddetti processi Markoviani. Ciascun elemento della catena è chiamato stato e lo stato successivo a quello attuale dipende esclusivamente da quest'ultimo e non da come si è giunti ad esso.

Processi stocastici

Un processo di tipo stocastico, per definizione, è un sistema dinamico con dinamiche di tipo stocastico (almeno parzialmente casuali).
Per ogni istante $t \in [0,\infty)$ il sistema si trova in uno stato *Xt*, preso da un insieme *S* (intervallo di stati).
Un processo di questo tipo è spesso scritto nella forma

$$X = \{ Xt \mid t \in [0,\infty) \}$$

Conseguenze

1. Si può parlare esclusivamente delle probabilità di trovare il sistema in stati particolari in tempi determinati: ogni stato *Xt* è una variabile casuale.
2. Per definire completamente un processo è necessario specificare le probabilità per lo stato *Xt* per ogni istante *t*.
3. Se il tempo in questione è un valore discreto (costituito da elementi isolati, non contigui tra loro)[4] si etichettano gli istanti di tempo attraverso dei numeri interi $n \geq 0$, scrivendo $X = \{ Xn : n \geq 0 \}$.

Processi Markoviani

Un processo di Markov è un processo stocastico (o aleatorio), ossia un insieme ordinato di variabili casuali, indicizzate dal parametro t *(tempo)*, nel quale **la probabilità di transizione** che determina il passaggio a un particolare stato di sistema **dipende unicamente dallo stato** di sistema **immediatamente precedente** e non dal come si è giunti a tale stato.
In poche parole ciascuna variabile aleatoria estratta all'istante t_n dipende soltanto dalla variabile estratta all'istante t_{n-1}.

Applicazioni di modelli di Markov

Questi modelli prendono forma all'interno di progetti in svariati ambiti, da quello medico sanitario alla scienza delle telecomunicazioni.

La **teoria delle code** che ne consegue trova applicazione in diverse situazioni: dalla fila alle poste ai pacchetti in coda in un router.

Formalmente questo può essere scritto come
$P(X(t_{n+1}) \leq x_{n+1} | X(t_n) = x_n, X(t_{n-1}) = x_{n-1}, \ldots, X(t_0) = x_0) = P(X(t_{n+1}) \leq x_{n+1} | X(t_n) = x_n)$
Questa è detta proprietà di Markov, o condizione di "assenza di memoria".
.

Proprietà di Markov

La proprietà di Markov, ossia il significato dell'aggettivo Markoviano, è un tipo di probabilità condizionata: la probabilità condizionata di un evento A rispetto a un evento B è la probabilità che si verifichi A, sapendo che B è verificato

$$P\left(x_{n+1} = j \mid x_n = i,\ x_{n-1} = i_{n-1},\ x_{n-2} = i_{n-2},\ \ldots,\ x_0 = i_0\right)$$

Dove:
$x_{n+1} = \ Stato\ Futuro$
$x_n = \ Stato\ Attuale$
$x_{n-1} = \ Stato\ Precedente$
$j, i \ = \ Stato\ Determinato$

"Qual è la probabilità che lo stato futuro x_{n+1} sia uguale ad un determinato stato j data l'intera storia passata dello stato attuale?"

Il processo di Markov semplifica tale definizione considerando solo l'informazione più recente:

$$P\left(x_{n+1} = j \mid x_n = i\right)$$

Ciò implica che:

1. la probabilità del futuro, dato passato e presente, dipende solo dal presente.
2. la probabilità del futuro, congiunta a quella del passato e conoscendo il presente, è

$$P\left(futuro \mid passato,\ presente\right) \ = P(futuro \mid presente) \ \cdot \ P(presente \mid passato)$$

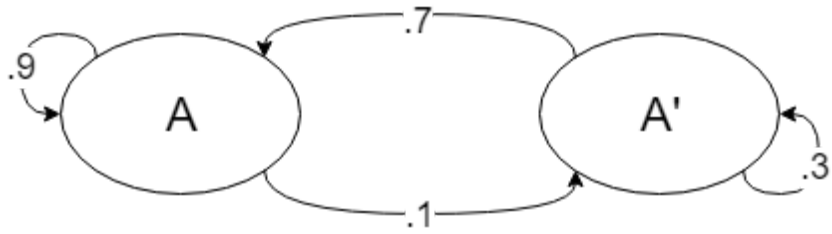
Gli eventi che soddisfano questa seconda condizione sono detti condizionalmente indipendenti, cioè sono indipendenti soltanto se vi è la conoscenza dello stato intermedio. Nel caso in cui non si conosca il presente, allora passato e futuro non sono più indipendenti.

Indicheremo con $P\left(x_{n+1} = j \mid x_n = i\right) \ = \ q_{ij}$ la probabilità di transizione.
Si tratta di un sistema omogeneo perché è indipendente dal tempo

Un esempio

L'impresa RedJuice, che produce succhi di frutta, possiede attualmente il **20%** dell'intera quota di mercato, e, al fine di incrementare il suo fatturato, incarica una società di marketing di prevedere le possibili variabili e i risultati di una nuova campagna pubblicitaria
Dai risultati dell'analisi, si conclude che in seguito alla campagna pubblicitaria, dopo una settimana i consumatori dei succhi RedJuice resteranno fedeli al brand con una probabilità del 90%, mentre chi è attualmente cliente di altri brand competitrici, inizierà a consumare succhi RedJuice con una probabilità del 70%.

Indichiamo con A l'impresa RedJuice e con A' tutti gli altri competitori.



$Q = (q_{ij})$ può essere scritto come una matrice, chiamata **matrice di transizione**:

$$\begin{pmatrix} .9 & .1 \\ .7 & .3 \end{pmatrix}$$

Ogni riga si riferisce a uno stato e ogni colonna allo stato al quale si vuole passare. ella cella indicata dalle due coordinate si inserisce la probabilità di passare dallo stato indicato dalla riga a quello indicato dalla colonna.

La somma delle probabilità di ogni riga nella matrice è sempre 1.

Per calcolare la probabilità che dopo la campagna pubblicitaria qualcuno consumi i prodotti di RedJuice è necessario operare l'unione sui casi favorevoli ossia

$$P(A_{n+1}) = P(A|A_n) + P(A|A'_n) = (.2)(.9) + (.8)(.7) = .74$$

Dopo una settimana la situazione sarà la seguente:

$$S_1 = [\ .74,\ .26\ ]$$

Dove le due colonne rappresentano rispettivamente la quantità di clienti del brand A e dei brand competitori A' dopo una settimana.

Lo stesso risultato è ottenibile moltiplicando le due matrici

$$S_{n+1} = \begin{bmatrix} .2 & .8 \end{bmatrix} * \begin{bmatrix} .9 & .1 \\ .7 & .3 \end{bmatrix}$$

e così via fino ad ottenere lo stato S_i che ci interessa

La matrice di transizione può essere utilizzata per determinare, ad esempio, la probabilità di passare da uno stato i a uno stato j in n steps.
Supposto all'istante n *(attuale)*, la catena X_n ha distribuzione $\bar{s}$ *(funzione di massa di probabilità indicata in forma vettoriale* $I \times M$*)*.

$$P(X_{n+1} = j) = \sum_i P(X_{n+1} = j \mid X_n = i) \cdot P(X_n = i)$$

E si legge: *"La probabilità che lo stato futuro* X_{n+1} *sia uguale ad un determinato stato* j *equivale alla sommatoria su tutti gli stati* i *delle probabilità che lo stato futuro* X_{n+1} *sia uguale ad un determinato stato* j *dato lo stato attuale* X_n *equivalente allo stato determinato* i *per la probabilità che lo stato attuale* X_n *sia uguale ad un determinato stato* i*".*

Riscrivibile nella forma

$$P(X_{n+1} = j) = \sum_i q_{ij} \cdot s_i$$

Dove q_{ij} è la probabilità di determinare uno stato futuro dato lo stato attuale e s_i è la funzione di massa (essa determina la probabilità di un evento X di assumere il valore di un determinato evento x) di probabilità dello stato attuale.

INFORMATICA

In ambito informatico può sembrare che il caso non ricopra alcun ruolo, tuttavia uno dei principali limiti con cui ancora oggi i sistemi informatici hanno a che fare è la generazione di numeri casuali. Tale necessità è richiesta in diverse applicazioni: nella sicurezza informatica e quindi negli algoritmi di crittografia che richiedono che i valori utilizzati siano imprevedibili e frutto di una generazione completamente casuale. D'altro canto, anche una semplice *Slot Machine* necessita di ciò per far si che qualsiasi giocata sia imprevedibile, o, ad esempio, che le estrazioni di una lotteria online siano puramente randomizzate.
Quindi è necessario porsi la seguente domanda:

Può un calcolatore generare un numero puramente casuale?

Generatori di numeri casuali

Un **random-number generator (RNG)**[5] è un dispositivo, sia fisico che computazionale, utilizzato per generare una sequenza di numeri che sia **impossibile da prevedere.**
Generalmente, nelle applicazioni dove la randomizzazione ricopre un ruolo primario, come nella crittografia, vengono utilizzati dei generatori di tipo hardware piuttosto che l'utilizzo di algoritmi pseudo-casuali.

Cos'è un numero casuale?

Un numero si definisce casuale se, preso da un insieme, ha la stessa probabilità di uscita di un qualsiasi altro numero di tale insieme ossia è equamente distribuito.
Ciascun numero dell'insieme considerato deve essere statisticamente indipendente dai restanti.

Numeri “veri” e numeri pseudo-casuali

Esistono due approcci per la generazione di un numero o una sequenza di numeri casuali.

Il primo misura alcuni **fenomeni fisici** considerati casuali eliminando ogni forma di prevenzione durante il processo di misurazione. Alcune tecniche consistono in misurare il rumore atmosferico, il rumore termico ed ulteriori fenomeni di tipo elettromagnetico e quantistico. Ad esempio, misurando l’entropia di fenomeni quali la radiazione cosmica di fondo o il decadimento radioattivo. Un problema solito di questo metodo è la bassa velocità dovuta quando il livello di entropia non è sufficientemente alto e porta il sistema a bloccarsi.

CURIOSITA’

A partire dal chipset *Intel i810*, le schede madri montano un chip che utilizza il rumore termico per generare dei *bits* casuali considerati imprevedibili quantisticamente.

Il secondo metodo, invece, utilizza **algoritmi computazionali** che si servono di formule matematiche o tabelle pre-calcolate in grado di produrre lunghe sequenze di numeri **apparentemente casuali** che in realtà sono determinati da un valore iniziale di grandezza notevolmente inferiore conosciuto come seme o chiave.
Un importante conseguenza di ciò è che l’intera sequenza, apparentemente casuale, **può essere riprodotta se si conosce la chiave**. Questi algoritmi vengono chiamati generatori di numeri pseudo-casuali (PRNG)[6].

Il risultato di una generazione pseudo-casuale di 0 e 1 nel tempo.

Tra i più conosciuti si può trovare il **generatore lineare congruenziale**:

$$X_{n+1} = (aX_n + c) \ mod \ m$$

Dove

X_{n+1} è la sequenza di valori pseudo-casuali
a è il moltiplicatore
c è l’incremento
X_n è il seme o chiave

Generatori di numeri pseudo-casuali

Come descritto precedentemente, questa tipologia consiste in algoritmi che utilizzano particolari formule matematiche.
Anche se un generatore di numeri pseudo-casuali non potrà mai produrre numeri realmente casuali, in pratica, nella maggior parte dei casi, è sufficiente per garantire un livello di sicurezza minimo nelle sue implementazioni.

Bisogna però ricordare che non sono algoritmi perfetti, come dimostra l'esempio seguente:
Se si utilizza la funzione **rand()** del linguaggio **PHP** su sistemi GNU/Linux, le probabilità di ottenere un risultato soddisfacente, nella generazione di numeri casuali, saranno alte. Al contrario, utilizzando lo stesso linguaggio su sistemi Microsoft Windows ci si può accorgere di come i numeri generati non siano così casuali come ci si potrebbe invece aspettare.

Un esempio

Nell'aprile 2008, in uno studio[8] effettuato da Bo Allen, sono stati confrontati i risultati di uno script PHP per generare numeri casuali lanciato sia su un sistema GNU/Linux che su un sistema Microsoft Windows. I risultati sono stati poi rappresentati in forma grafica utilizzando una libreria grafica chiamata **GD**.

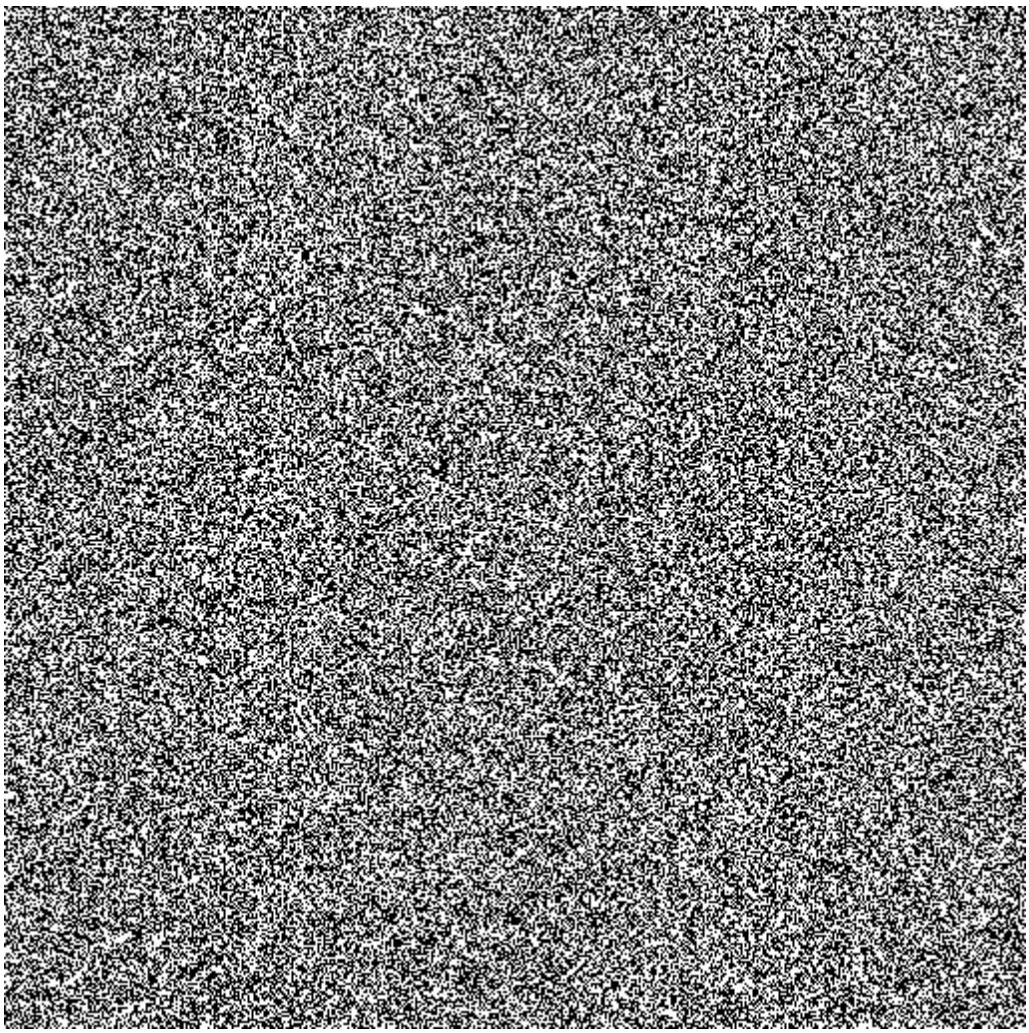
Lo script utilizzato è il seguente:

```
<?php
// Richiede la libreria GD
header("Content-type: image/png");
$im = imagecreatetruecolor(512, 512)
    or die("Cannot Initialize new GD image stream");
$white = imagecolorallocate($im, 255, 255, 255);
for ($y = 0; $y < 512; $y++) {
    for ($x = 0; $x < 512; $x++) {
        if (rand(0, 1)) {
            imagesetpixel($im, $x, $y, $white);
        }
    }
}
imagepng($im);
imagedestroy($im);
```

I risultati, come è possibile constatare nella pagina successiva, sono stati decisamente diversi e molto interessanti.

Generazione su sistemi Linux

L’immagine seguente è il risultato grafico dello script PHP mostrato precedentemente utilizzato su un sistema di tipo GNU/Linux.

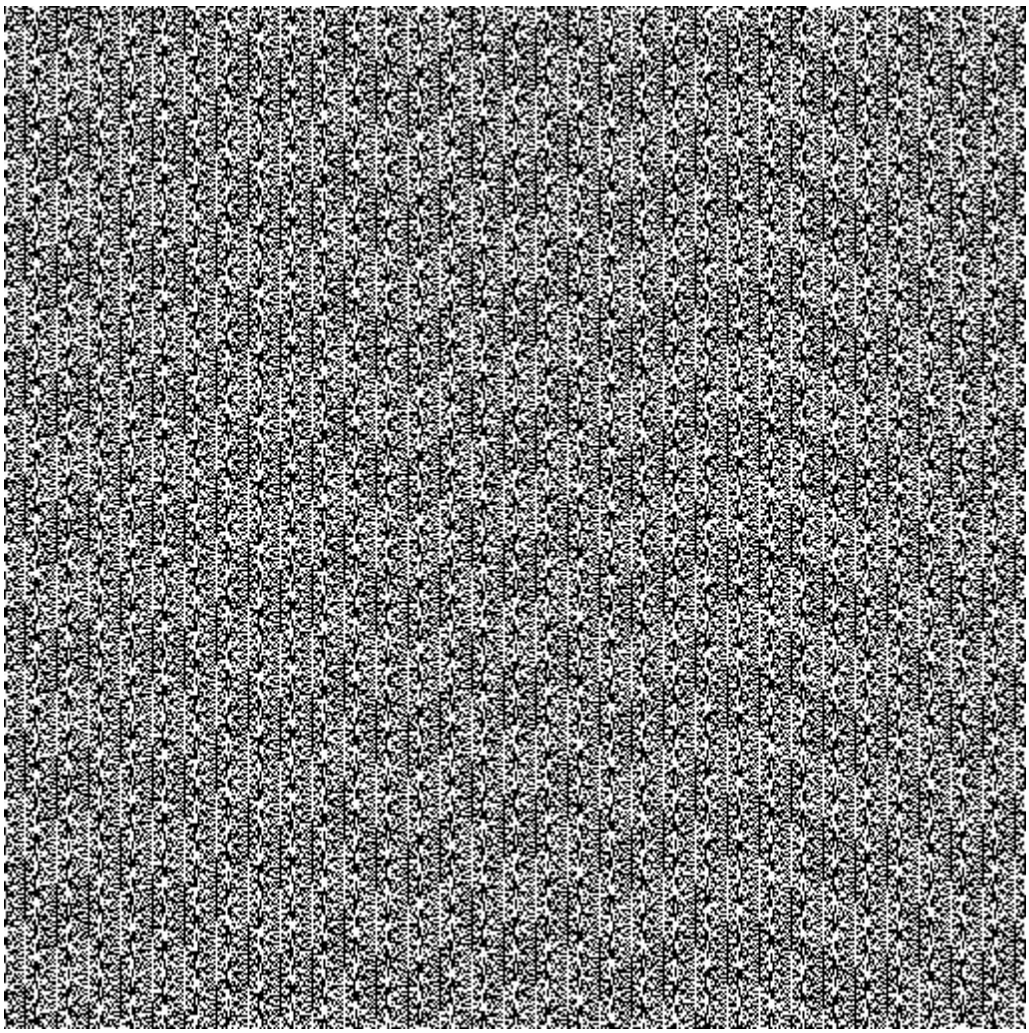


Rappresentazione grafica di valori generati con PHP su Linux

Il risultato si può considerare soddisfacente in quanto l’immagine prodotta non sembra seguire alcun pattern.

Generazione su sistemi Windows

L’immagine seguente è invece il risultato grafico della produzione di una sequenza di numeri casuali generati utilizzando il linguaggio PHP su un sistema Microsoft Windows



Rappresentazione grafica di valori generati con PHP su Windows

È facile notare come la distribuzione di tali non sia puramente casuale ma sembri seguire un pattern specifico. È teoricamente impossibile provare che un generatore di numeri pseudo-casuali sia veramente casuale.

Generatori di numeri puramente casuali

Differentemente dai generatori descritti precedentemente, i *True Random Number Generators*[7] (**TRNG**) si servono di dati estratti da **fenomeni fisici** come il movimento del mouse o gli intervalli di tempo tra la pressione di un tasto e l'altro. Tuttavia è importante prestare attenzione a quale metodo si utilizza. Ad esempio, l'intervallo di tempo tra i diversi tasti premuti può essere influenzato dal fatto che, spesso, i sistemi operativi bufferizzano gli input da tastiera prima di inviarli al programma interessato che interpreterà i tasti premuti come se fossero stati premuti contemporaneamente.
Tuttavia, esistono molti altri modi per ottenere valori veramente casuali; uno di questi è quello di utilizzare una fonte radioattiva: il tempo necessario ad un processo di decadimento radioattivo (la trasformazione di una particella elementare in uno o più oggetti differenti) è imprevedibile e questo dato può essere utilizzato dai calcolatori per servire questa richiesta. Il servizio chiamato *HotBits* del sito Fourmilab è un esempio di generatore di numeri casuali che utilizza questa tecnica.

CURIOSITA'

Un altro approccio, utilizzato da siti come *random.org*, è quello di sfruttare il rumore atmosferico che è possibile ricavare utilizzando una semplicissima radio.

Confronto tra *PRNGs* e *TRNGs*

Vi sono varie differenze l'ultima tipologia di generatori descritta e quella di numeri pseudo-casuali. La prima è che quest'ultimi sono inefficienti a causa del maggior tempo necessario all'ottenimento di un risultato. Inoltre si tratta di metodi **indeterministici**, ossia che, data una sequenza di numeri generati, essa non è riproducibile perché il dato di partenza non è prevedibile né individuabile.

Caratteristiche	**Generatori di numeri pseudo-casuali**	**Generatori di numeri puramente casuali**
Efficienza	Eccellente	Scarsa
Determinismo	Deterministici	Non deterministici
Periodicità	Periodici	Aperiodici

Generazione di testi casuali

"Mostratemi i vostri diagrammi di flusso e nascondetemi le vostre tabelle e continuerò a essere confuso. Mostratemi le vostre tabelle e non avrò più bisogno dei vostri diagrammi di flusso; saranno ovvi."
Frederick P. Brooks, Jr. *The Mythical Man-Month*

La definizione di strutture dati è fondamentale al fine di semplificare la struttura del codice.
L'essenza del problema analizzato è tipica di moltissimi programmi: elaborare dei dati in ingresso per ottenere dei dati in uscita.
Più precisamente, si vuol generare del testo casuale che rispetti le regole sintattiche di un linguaggio, ma non obbligatoriamente corretto lessicalmente o morfologicamente. Generando lettere o parole casuali, il risultato non sarebbe corretto.
Per esempio, un programma che selezioni a caso delle lettere (e degli spazi per separare le parole) genererebbe in output una stringa del tipo:

gehtd brasdf piaqhzlydc ldcygtu pasdfz zufbn lqpdfmnyte ayvakhvy

che non ha alcun senso e non rispetta alcuna regola grammaticale. Bilanciando invece la frequenza di comparsa delle lettere (prendendo in esempio la lingua inglese), si otterrebbe un risultato del tipo:

idtefoae tcs trder jcii ofdslnqetacp t ola

che tuttavia non convince ancora. Allo stesso modo, parole scelte casualmente all'interno di un dizionario non sono in grado di costruire un testo simile ad uno reale:

andesite overwetly zetes emancipating epicurean

Al fine di ottenere risultati più convincenti è necessario ricorrere ad un modello statistico più strutturato, per esempio la frequenza di apparizione di intere frasi o di parole al loro interno. Ma come è possibile ricavare tali statistiche senza dover obbligatoriamente studiare una buona base della lingua con la quale si vuole produrre il testo?
La soluzione è quella di costruire un modello statistico del linguaggio così come viene usato in un testo qualsiasi, e, partendo da questo, generare del testo casuale con proprietà statistiche simili all'originale.

Le catene di Markov in informatica

Un'implementazione[9] elegante di questa tecnica è data dall'algoritmo delle catene di Markov.
Se si immagina l'input come una sequenza di frasi subordinate, l'algoritmo divide ogni frase in due parti: un ***prefisso*** formato da *n* parole e un ***suffisso***, formato da una sola parola, che segue il *prefisso.* L'algoritmo in questione, dopo aver eseguito una mappatura del testo di partenza, produce in output delle frasi mediante la scelta casuale del suffisso che segue il prefisso, in base alle proprietà statistiche del testo originale. Frasi di tre parole funzionano bene. Un prefisso di due parole viene usato per selezionare la parola di suffisso:

```
imposta w1 e w2 alle prime due parole nel testo
    stampa w1 e w2
    ciclo:
        scegli casualmente w3, uno dei successori del prefisso w1, w2 nel testo
        stampa w3
        sostituisci w1 e w2 con w2 e w3
        ripeti il ciclo
```

Ad esempio, si vuole generare del testo casuale partendo dal testo seguente, utilizzando prefissi di due parole:

Show your flowcharts and conceal your tables and I will be mystified.
Show your tables and your flowcharts will be obvious . (end)

Queste sono alcune delle coppie di parole di input e delle parole che le seguono:

Prefisso di Input:	**Parole del suffisso che seguono**
Show your	flowcharts tables
your flowcharts	and will
flowcharts and	conceal
flowcharts will	be
your tables	and and
will be	mystified. obvious.
be mystified.	Show
be obvious.	(end)

Un algoritmo di Markov che elabora questo testo inizierà stampando Show your e poi selezionerà a caso flowcharts o tables. Se sceglie il primo suffisso, il prefisso diventa your flowcharts e la prossima parola sarà and o will. Se invece sceglie tables la prossima parola sarà and. Questo processo si ripete finché non si incontra l'indicatore di terminazione. Il numero di parole presenti nel prefisso, due nell'esempio in esame, è un parametro che, se ridotto, porta l'algoritmo a produrre prosa meno coerente; al contrario, aumentandolo, si tende a riprodurre il testo di input alla lettera. Per testi in lingua inglese, ad esempio, un prefisso formato da due parole è un buon compromesso.
Seppur una parola sia considerata come una sequenza di caratteri alfabetici, è consigliabile lasciare la punteggiatura legata alle parole: "words" e "words." sono considerate due parole differenti. Ciò permetterà che punteggiatura e grammatica influenzino la scelta della parola, sebbene permettano anche alle virgolette e alle parentesi non chiuse di introdursi.

Migliorare le prestazioni

Se si volessero utilizzare testi in input di molte parole, ad esempio $n = 100.000$ per garantire prestazioni elevate è necessario rivedere gli algoritmi e le strutture dati utilizzate.
Il programma deve prima salvarsi l'intero input prima di elaborarlo e produrre l'output. Un metodo potrebbe essere quello di salvarsi l'input sotto forma di array di puntatori alle singole parole; per produrre ogni parola, sarebbe dunque necessario scorrere tutto il testo per trovare tutti i suffissi che possono essere accodati ad essa. Questo significa analizzare tutte le 100.000 parole ad ogni parola ossia effettuare n^2 operazioni, processo che richiederebbe troppo tempo. È necessaria una struttura dati che meglio rappresenti ogni prefisso e i rispettivi suffissi.
Gli steps del programma sono i seguenti:

1. Leggere l'input costruendo le strutture dati necessarie.
2. Produrre l'output servendosi delle strutture dati costruite.

In entrambe le fasi è necessario individuare velocemente ogni prefisso: nella fase di input per aggiornare i corrispettivi suffissi, e nella fase di output per selezionare in maniera casuale il prefisso corrispondente.
L'utilizzo di una *hashmap* dove le chiavi sono i prefissi e i valori i suffissi, facilita e velocizza il processo.

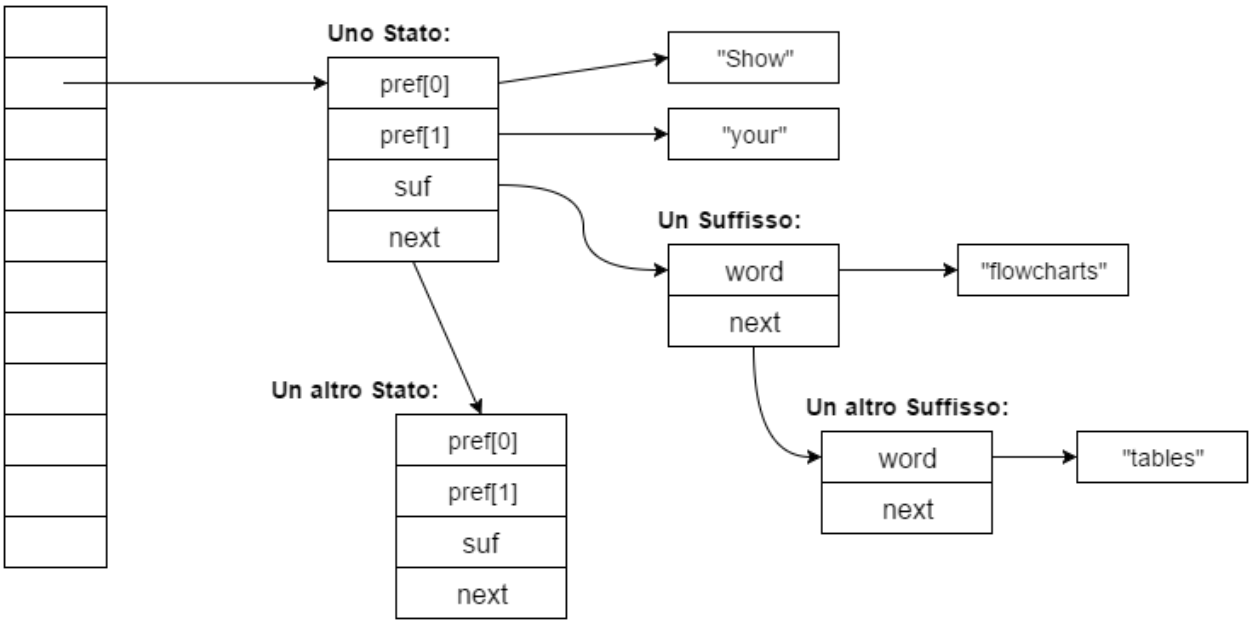
CURIOSITA'

L'ordine di complessità per operazioni *get* e *put* nelle hashmap è di O(1)

L’implementazione proposta considera prefissi formati da due parole. Di conseguenza, ciascuna parola generata in output dipenderà dal prefisso di parola che la precede.

Durante la fase di lettura dell’input bisogna salvare tutte le parole che seguono un dato prefisso; non conoscendo il loro numero è bene utilizza una lista o un array dinamico in modo da garantire velocità ed efficienza man mano che vengono riempite.

Schema di funzionamento



Schema di funzionamento dell’algoritmo delle catene di Markov

SISTEMI E RETI

"Chiunque consideri i metodi aritmetici metodi di produrre numeri casuali è, naturalmente, in uno stato di colpa."
John Von Neumann

Il caso e la crittografia

Come citato precedentemente, la generazione di numeri casuali è un processo di fondamentale importanza nella **crittografia**, ma il vero motivo qual è?
La spiegazione è semplice, bisogna rendere impossibile risalire alla chiave di cifratura utilizzata in un determinato contesto; per garantire ciò è necessario che essa sia frutto di una generazione casuale, non deterministica, di un processo irreversibile.

Un esempio pratico

Nel 2009 è stato effettuato un **attacco**[10] al sito web *Hacker News* da parte di un utente sotto il permesso del proprietario del sito stesso.
Ecco come è successo:
Quando si accede ad un sito, solitamente viene assegnato un ID ad ogni sessione (finché non viene chiuso il browser). Tale ID è unico e non deve essere ottenibile da nessun altro, che altrimenti, potrebbe impersonare il vero "proprietario" di quel determinato ID.
Ogni ID è generato tramite un generatore di numeri pseudo casuali, che, se non basati su algoritmi sufficientemente sofisticati, possono comportare seri problemi a livello di sicurezza.
In questo caso il seme, o chiave, veniva generato partendo dal tempo espresso in millisecondi in cui veniva effettuata la richiesta, sul quale venivano successivamente eseguite alcune operazioni non sufficientemente complesse per rendere impossibile la ricostruzione dell'ID.

Un esempio teorico

L'esempio appena mostrato non riguarda direttamente la crittografia, ma un importante aspetto della sicurezza. Tuttavia i numeri casuali hanno un'importante ruolo anche nella crittografia.

Ad esempio, ogni sessione **HTTPS** inizia in questo modo:

1. Il browser invia diverse informazioni al server tra cui la versione SSL che vuole utilizzare.
2. Il web server risponde con informazioni simili riguardanti la versione SSL e il proprio certificato.
3. Il browser verifica l'autenticità del certificato e genera un codice segreto che sarà utilizzato per rendere sicura la connessione che utilizzerà poi la cifratura simmetrica basata sul codice segreto per rendere sicuri i dati trasmessi.

Per questo motivo **deve essere impossibile risalire al segreto condiviso** tra browser e server.

CURIOSITA'

Verificare l'autenticità del certificato può prevenire attacchi come il "Man in the middle". Esso consiste in un'intrusione nella quale il malintenzionato finge di essere il destinatario della comunicazione inviando la propria chiave pubblica al momento della connessione.

Un altro esempio è dato da una parte del processo di connessione di un Client ad un Access Point Wi-Fi utilizzando il protocollo **WPA-2**.

1. L'Access Point invia un *nonce* (un valore generato casualmente) al Client che lo utilizzerà per costruire la PTK (Pairwise Transient Key).
2. Il Client invia un *nonce* (e un MIC) all'Access Point che lo utilizzerà per costruire la PTK.

Il Client e l'Access Point utilizzeranno questi due valori scambiati per rendere sicure tutte le comunicazioni successive.

Come è intuibile, in tutti questi processi è fondamentale che i valori iniziali sui quali si basa la sicurezza della comunicazione siano imprevedibili e che sia impossibile risalirvi in alcun modo.

CURIOSITA'

Già nel 1976, l'algoritmo sviluppato da Diffie e Hellman per lo scambio di codice segreto su un canale non sicuro si basava su un segreto condiviso ottenuto partendo da due valori condivisi (base e modulo) combinati con due numeri generati casualmente.

LETTERATURA

«Le cose tutte quante hanno ordine tra loro, e questo è forma che l'universo a Dio fa simigliante.»
D.Alighieri, *La Divina Commedia (Paradiso, I)*

Fin dall'antichità il tema del caso è stato affrontato in numerosissime opere ed è stato in grado di condizionare ideologie e studi. Ne sono state date diverse interpretazioni come quella di destino o di **fortuna**. Quest'ultima, secondo l'aspetto filosofico, è una specificazione del caso e, precisamente, è il caso, in quanto può recare all'uomo vantaggi e danni.
Identificabile con la Tyche, col fato, con la Provvidenza, con il destino, quindi, a seconda delle epoche e delle concezioni, essa assume varie sfaccettature e significati.

La fortuna nella Divina Commedia

Nella Divina Commedia[11] di Dante Alighieri (1265-1321) il caso, inteso come fortuna o provvidenza, è un tema ricorrente.

In particolare nel ***VII* Canto** dell'**Inferno**, una volta giunti nel V cerchio, ospitante le anime di coloro che in vita hanno avuto un'immoderata brama delle ricchezze, ossia avari e prodighi, Virgilio afferma che i beni terreni, affidati alla fortuna, sono effimeri e tutto l'oro del mondo sarebbe insufficiente a placare quelle anime afflitte. Dante, perplesso, chiede più dettagli a riguardo e il suo maestro dapprima biasima l'ignoranza del mondo, quindi spiega che Dio ha disposto diverse intelligenze angeliche a governare i vari Cieli, e allo stesso modo ha creato un'intelligenza che amministri i beni terreni. Essa, la fortuna (impersonata dalla dea Tiche), stabilisce quando le ricchezze debbano cambiare di mano e quali genti debbano prosperare o decadere, secondo l'imperscrutabile giudizio divino. L'uomo non comprende il senso del variare delle fortune umane, che risponde però ad una logica superiore. Molti sciocchi la maledicono, mentre dovrebbero lodarla e ringraziarla: essa non sente neppure queste lamentele, gira la sua ruota e opera serenamente insieme agli altri angeli.

Per Dante e per la cultura cristiana in genere, la fortuna per certi aspetti si identifica con la Provvidenza: è infatti un'intelligenza celeste, destinata da Dio al governo delle cose del mondo e distributrice dei beni secondo il suo volere imperscrutabile.

Nella concezione cristiana non esiste il fato, ma la provvidenza. L'uomo, con le sue forze, non può opporsi al fato e quindi è portato alla rassegnazione di fronte ad avvenimenti che non riesce a determinare. Con l'avvento del cristianesimo al concetto di fato si sostituisce quello di Provvidenza, cioè l'ordine con il quale Dio regola il creato e determina lo sviluppo della storia. La visione provvidenziale della vita è rasserenante per il credente. Nulla infatti è lasciato al caso, disgrazie e dolori, fortune e successi sono distribuiti agli uomini secondo un disegno ben determinato e che ha come fine il bene dell'umanità.

Ciò è constatabile nel **Canto I** del **Paradiso:** durante l'ascesa di Dante e Beatrice al Paradiso, il protagonista ha diversi dubbi, tra cui come sia possibile che lui, dotato di un corpo mortale, sia in grado di salire oltre l'aria e il fuoco. Beatrice spiega infatti che tutte le cose sono state ordinate secondo la **provvidenza divina**, così che ogni cosa tenda al suo fine attraverso strade diverse. Ciò vale per le cose inanimate, come il fuoco che tende a salire verso l'alto per sua natura ma anche per gli esseri intelligenti, la cui anima razionale tende naturalmente a muoversi verso Dio; ovviamente essi sono dotati di libero arbitrio, per cui può avvenire che anziché volgersi in quella direzione siano attratti dai beni terreni, ma questo non è il caso di Dante che ha ormai purificato la sua anima nel viaggio attraverso Inferno e Purgatorio. Egli tende dunque verso Dio che risiede nell'Empireo e ciò è un atto del tutto naturale, come quello di un fiume che scorre dall'alto verso il basso, mentre sarebbe innaturale per Dante restare a terra, come un fuoco la cui fiamma non tendesse verso l'alto.

KANJI APP

KanjiApp è un motore di ricerca per caratteri giapponesi chiamati Kanji. Esso integra anche una funzione per generare testi casuali implementando l'algoritmo delle catene di Markov.

Tecnologie utilizzate - Front End

Le tecnologie front end denotano la *user interface*, ossia la parte visibile all'utente del programma con la quale esso può interagire. Quelle utilizzate per la costruzione di KanjiApp sono: HTML5, CSS3 e AngularJS.

HTML5

L'Hyper Text Markup Language[12] è il principale linguaggio di markup utilizzato per la formattazione e l'impaginazione di pagine web. Esso è un linguaggio di pubblico dominio nato nel 1993 presso il CERN di Ginevra e la sua sintassi è stabilita dal World Wide Web Consortium. I browser interpretano il contenuto di una pagina HTML in base ai tag presenti. Ciascuna porzione di contenuto è racchiusa tra due tag, uno di apertura e uno di chiusura, e ogni tag è caratterizzato da una parola chiave racchiusa tra due apici (es. "<h1> titolo </h1>").

CSS3

CSS3[13] è il più recente standard CSS (**Cascade Style Sheets**), ossia un linguaggio finalizzato alla definizione della formattazione delle pagine HTML, XHTML e XML. Questo linguaggio consente la definizione di classi e id al fine di migliorare l'organizzazione generale della pagina.
La definizione di stili css può essere di tre tipi: **in linea**, **nell'head** o **esterna**.
La prima viene eseguita direttamente all'interno del tag interessato utilizzando la direttiva "style"; viene utilizzata raramente, nel caso bisogni dichiarare un basso numero di attributi.
La dichiarazione nell'head, come è possibile intuire dal nome, viene eseguita all'inizio della pagina, all'interno del tag <*head*>.
L'ultima è la più frequente, ossia esterna: consiste nell'utilizzare un file esterno nel quale inserire gli attributi necessari a modificare l'aspetto della nostra pagina. È necessario *linkare* il file all'interno dell'*head* servendosi del tag <*link*>.

AngularJS

AngularJS[14] è un **framework** utilizzato costruire applicazioni web dinamiche. Permette di utilizzare l'HTML come linguaggio di *template* estendendolo con la definizione di componenti aggiuntivi. Il *data binding* e il *dependency injection* consentono di eliminare grosse porzioni di codice che altrimenti si dovrebbero scrivere.

Tecnologie utilizzate - Back End

L'insieme delle tecnologie back end si occupa dell'effettiva elaborazione dei dati forniti dal front end. La distinzione di una parte di ingresso e di una parte terminale nei sistemi software è un genere di astrazione che aiuta a mantenere le diverse parti di un sistema complesso logicamente separate e quindi più semplici. Per la costruzione di "KanjiApp" sono stati utilizzati *Apache*, *PHP5, Unirest* e *Java*.

Apache

Apache[15] è la più diffusa piattaforma **server Web** modulare. La sua funzione principale è quella di realizzare le funzioni di trasporto delle informazioni, di internetwork e di collegamento, ed ha il vantaggio di offrire funzioni di controllo per la sicurezza come quelle effettuate da un proxy.

PHP 5

PHP[16], originariamente chiamato **"Personal Home Page"**, è un linguaggio di scripting interpretato nato per la programmazione di pagine web dinamiche e utilizzato principalmente per la programmazione lato server.

Unirest

Al fine di utilizzare le API che descritte successivamente, si utilizza [17]Unirest: un insieme di librerie HTTP, create e mantenute da mashape, disponibili in diversi linguaggi di programmazione (in questo caso PHP).

Java

Java[18] è uno dei più famosi linguaggi di programmazione **orientati agli oggetti** e progettato per essere il più possibile indipendente dalla piattaforma di esecuzione. Il codice compilato che viene eseguito su una piattaforma, infatti, non deve essere ricompilato per essere eseguito su una piattaforma diversa. Il prodotto della compilazione è infatti in un formato chiamato *bytecode* che può essere eseguito da una qualunque implementazione di un processore virtuale detto **Java Virtual Machine**.

Tecnologie e risorse generali

Tra le risorse e le tecnologie generali utilizzate per la creazione di “KanjiApp” vi sono alcune API gratuite disponibili su *mashape*, *Cloud9* e *git.*

Mashape API

Mashape[19] è un marketplace di API (**Application Program Interface**) che mette a disposizione degli utenti API sia gratuite che a pagamento, situtate sul cloud.
In particolare, per questa applicazione, è stata utilizzata un’ API fornita da “KanjiAlive” chiamata “Learn to read and write Japanese kanji”[20]. I risultati sono ottenuti in formato **JSON** attraverso l’utilizzo del protocollo **HTTPS**.

Cloud9

Cloud9[21] è l’**ambiente di sviluppo online** utilizzato per la scrittura di questa web app. Esso è basato su ubuntu e supporta oltre 40 linguaggi. La comodità derivante dall’utilizzo di questo servizio è data dalla possibilità di accedere facilmente al proprio codice, sempre aggiornato all’ultima modifica effettuata, da qualsiasi computer e di poterci lavorare anche simultaneamente con altri utenti. Ciò, tuttavia, comporta lo svantaggio di richiedere una connessione internet per il suo utilizzo, che non sempre è disponibile.

Git

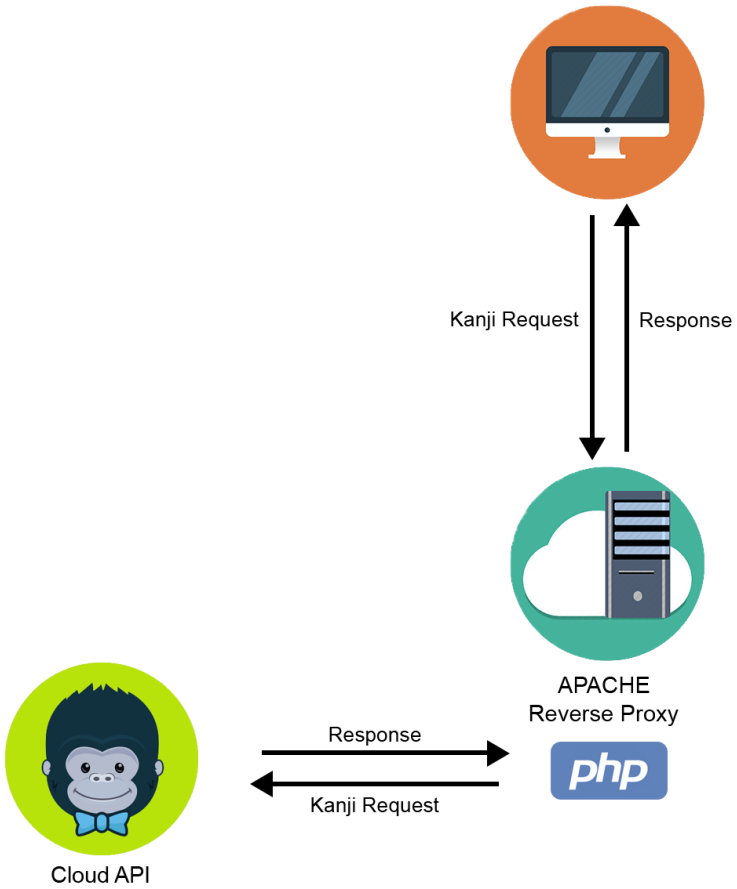
Git[22] è un software open source di **controllo di versione distribuito** (DVCS) utilizzabile tramite interfaccia a linea di comando creato da Linus Torvalds nel 2005.
Esso permette di tenere traccia delle modifiche e delle versioni apportate al codice sorgente di un software, senza la necessità di dover utilizzare un server centrale.
Con questo sistema gli sviluppatori possono collaborare individualmente e parallelamente da offline su di un proprio ramo (branch) di sviluppo, registrare le proprie modifiche (commit) ed in seguito condividerle con altri o unite (merge) a quelle di altri, il tutto senza bisogno del supporto di un server.

Schema di funzionamento

Come mostrato nelle tecnologie *back end* utilizzate, è stato necessario sia l'utilizzo di PHP che quello di Java. PHP viene utilizzato per le richieste relative ai *kanji* mentre Java viene impiegato per la parte del programma riguardante l'algoritmo delle catene di Markov.

Apache/PHP WebServer

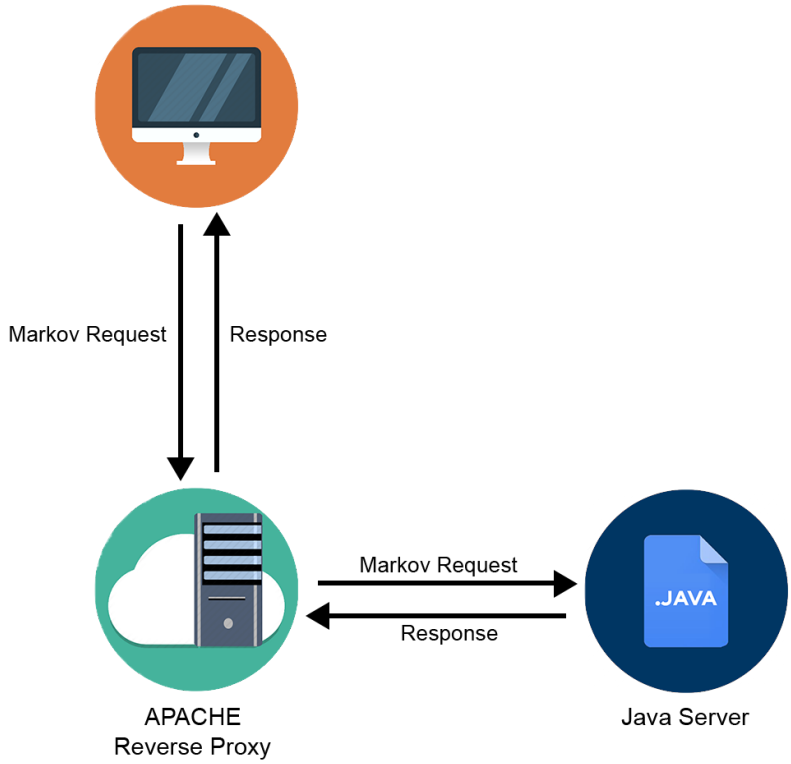
PHP è stato utilizzato inizialmente per poter effettuare le richieste HTTP all'API su *mashape.com*. È stato necessario l'impiego di un web server per ovviare alla restrizione imposta dal **Cross Origin Resource Sharing (CORS),** limitazioni imposte dai browser per motivi di sicurezza che limitano le richieste http inizializzate all'interno di uno script. Ad esempio, la funzione $http di *AngularJS* è basata sull'API XMLHttpRequest, che segue la *same-origin Policy;* in poche parole è necessario che l'origine della richiesta sia la stessa della risorsa alla quale è destinata.



Schema di funzionamento di una richiesta di tipo 'kanji'

Java Server

L'impiego di Java, invece, è stato necessario per l'implementazione dell'algoritmo delle catene di Markov, sia per motivi prestazionali che organizzativi.
Per utilizzare più web server in un unica applicazione si è scelto l'inserimento di un **Reverse Proxy**. Apache, in questo caso, inoltra le richieste su una porta differente, che sarà ascoltata dal server in Java, nel caso provengano da una determinata origine (in questo caso */api/…*).



Schema di funzionamento di una richiesta di tipo 'Markov'

Layout

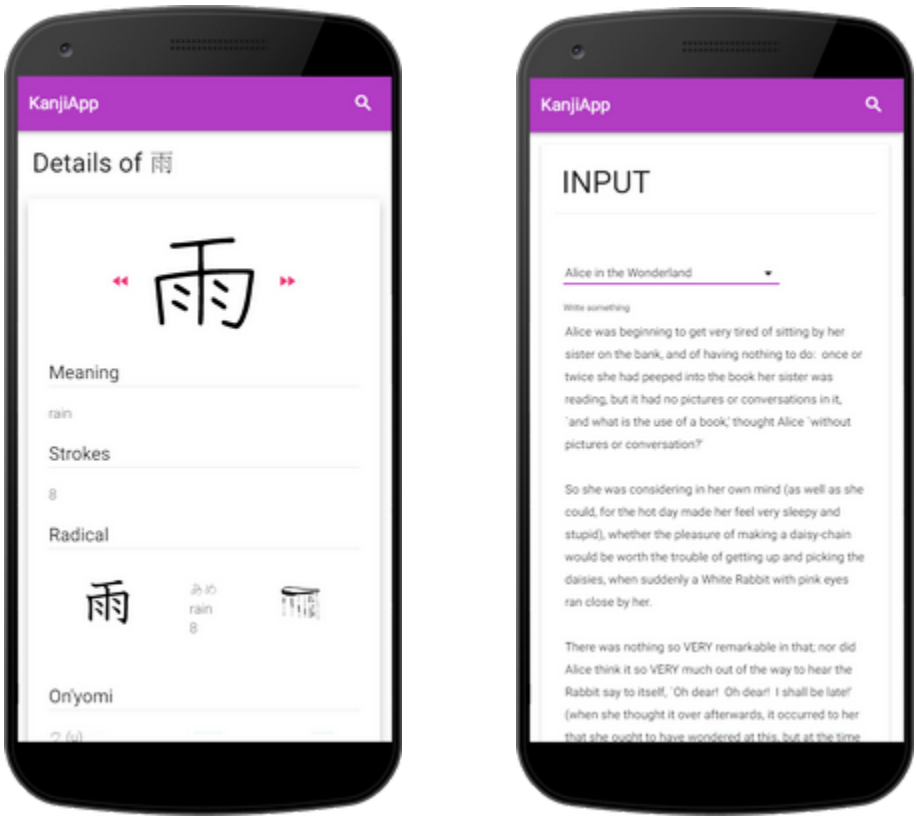
Material Design

L'intero design di "KanjiApp" è stato creato seguendo scrupolosamente le linee guida del **Material Design**[23], il più recente linguaggio visivo progettato da Google Inc. utilizzato prevalentemente sui dispositivi android, ma anche su moltissime web applications.
Per la sua implementazione si è scelto l'utilizzo dei componenti grafici di *Angular Material*[24], la cui combinazione con *AngularJS* consente il raggiungimento di prestazioni notevoli.

Inoltre si è scelto l'utilizzo delle "Material icons"[25] di Google Inc. al fine di rispettare al meglio le regole del Material Design.

Layout responsivo

Il layout è **completamente responsivo** ossia è in grado ad adattarsi in modo automatico al tipo di dispositivo sul quale viene visualizzata la web application: monitor, televisori, smartphones, tablets…



Anteprima dell'applicazione su smartphone

Utilizzo

Come descritto precedentemente “KanjiApp” svolge due funzioni:

- Motore di ricerca per i *kanji* (lo scopo per il quale è nata).
- Implementazione delle catene di Markov (funzionalità aggiunta in seguito).

La lingua giapponese

Il giapponese[26] è la lingua principale parlata in Giappone e in numerose aree di immigrazione giapponese. Essa è parte della famiglia delle lingue nipponiche.
In giapponese esistono cinque fonemi vocalici e ventisei fonemi consonantici differenti. Questi ultimi, però, non si presentano mai da soli ma hanno sempre bisogno di una vocale a cui appoggiarsi. Si usa dire a questo proposito che il giapponese sia una **lingua sillabica**: l'elemento fondamentale della parola non è infatti la lettera, ma la mora (composta secondo lo schema [consonante] + [vocale]).

Il sistema di scrittura si basa sue due *kana,* sui *kanji* e sul *romaji.*
I kana sono alfabeti sillabici e si dividono in *hiragana* e *katakana* mentre i *kanji* sono chiamati sinogrammi.
Il *romaji* è il sistema di traslitterazione dal giapponese ai caratteri latini.

CURIOSITA'

Il giapponese è parlato da oltre 127.000.000 persone, statistica che lo piazza al nono posto tra le lingue più parlate al mondo.

Hiragana

L'*hiragana* viene utilizzato per i prefissi, i suffissi e le particelle (posposizioni), ossia per quelle parti grammaticali che non sono rappresentabili con i *kanji.* Viene inoltre utilizzato per trascrivere la pronuncia dei kanji, nei testi informali o destinati ai bambini che ancora non hanno imparato molti dei *kanji*.

CURIOSITA'

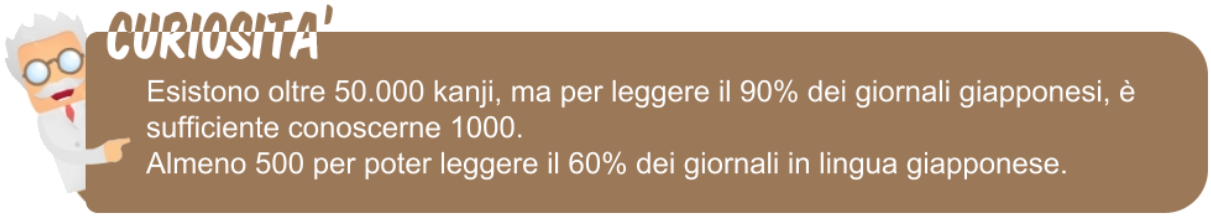
I cartelli stradali, in Giappone, presentano dei *kanji* con la loro rispettiva pronuncia in *hiragana*.

Katakana

Il *katakana* viene invece impiegato per la trascrizione di nomi e parole straniere

Kanji

I *kanji*[27] **derivano dalla scrittura cinese** e sono impiegati nella rappresentazione delle parti morfologicamente **invariabili** delle espressioni giapponesi. Infatti possono rappresentare le radici dei verbi, aggettivi e numerosi sostantivi.
Ad oggi presentano due tipologie di pronuncia: *on'yomi* e *kun'yomi*.
La prima, detta *on*, rappresenta la pronuncia cinese del *kanji*.
La *kun'yomi,* invece, rappresenta la pronuncia d'origine giapponese.



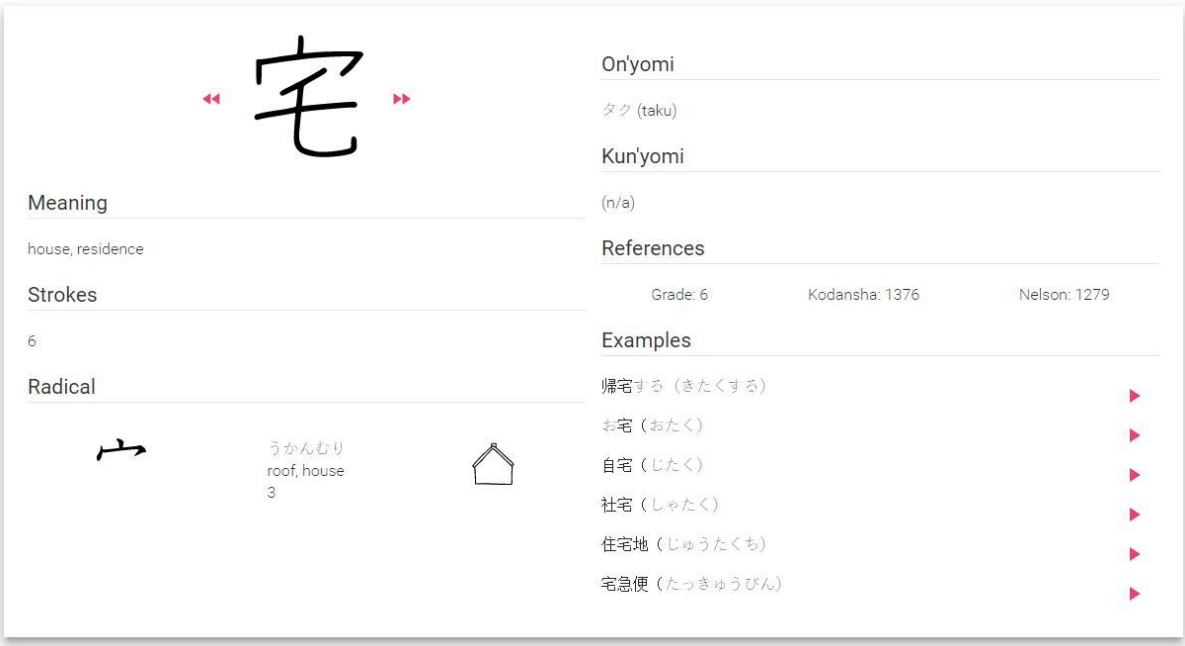
Come si usa *KanjiApp*?

Come già spiegato, una delle funzioni di questa applicazione è quella di ricercare un *kanji* partendo dal suo significato inglese o direttamente dal carattere stesso, per poi visualizzare numerosi dettagli a riguardo.

1. Cliccando sulla lente d'ingrandimento presente a destra della toolbar è possibile inserire il significato inglese del *kanji* cui si è interessati o il *kanji* stesso, nel caso già si conoscesse e si volessero visualizzare dei dettagli a riguardo.
2. Se la ricerca ha prodotto dei risultati, i rispettivi *kanji* saranno visualizzati in forma di tasselli.

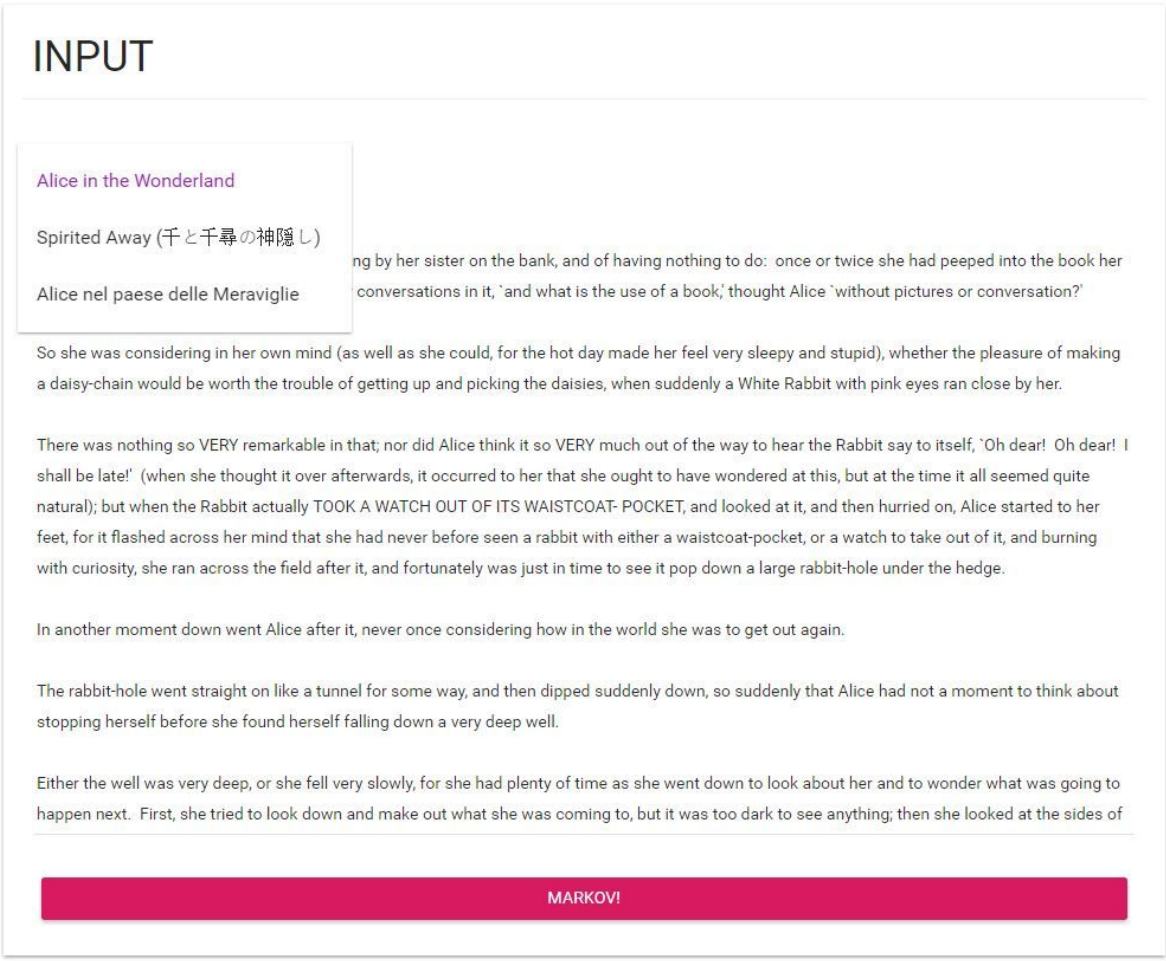
Esempio di ricerca con KanjiApp

3. Cliccando su uno dei risultati sarà possibile visualizzare una pagina contenente numerosi dettagli a riguardo tra cui:
    a. **Immagine** del kanji e **comandi** per visualizzarne l'intero processo di costruzione (realizzato tramite l'utilizzo di immagini vettoriali).
    b. **Meaning:** Significato in lingua inglese.
    c. **Strokes:** Numero di tratti necessari per la sua scrittura.
    d. **Radical:** Il carattere principale sul quale ciascun *kanji* è basato; ne esistono 214 nella lingua giapponese. Inoltre è possibile visualizzarne la pronuncia in *kun'yomi* e in *romaji*, il numero di tratti che lo compongono, il significato in lingua inglese e un suggerimento in forma grafica per favorirne l'apprendimento.
    e. **On'yomi:** La pronuncia cinese del *kanji* e la rispettiva pronuncia in *romaji*.
    f. **Kun'yomi:** La pronuncia giapponese del *kanji* e la rispettiva pronuncia in *romaji*.
    g. **References:** Il grado del *kanji* e il rispettivo indice all'interno dei dizionari *Kodansha* e *Nelson*.
    h. **Examples:** Una lista di esempi di utilizzo del *kanji* con la rispettiva pronuncia *kun'yomi* e *romaji*. Inoltre è possibile **ascoltare la pronuncia** dell'esempio scelto premendo sul pulsante *play* corrispettivo.



Esempio della visualizzazione dei dettagli di un kanji

La seconda funzionalità di *KanjiApp* è data dall'implementazione della **teoria delle catene di Markov**.
Il programma richiede l'inserimento di un testo di input, dal quale si vuole generare un testo casuale.
Per avere risultati soddisfacenti è necessario inserire un testo sufficientemente lungo; per ovviare a questo problema è stato aggiunto un menù a comparsa contenente alcuni esempi di testi (di cui uno in lingua giapponese).



Esempio di selezionamento di un testo per la generazione di un testo casuale

Una volta inserito il testo è sufficiente cliccare sul pulsante "Markov!" e l'applicazione mostrerà il risultato raggiunto. Ogni volta che il pulsante viene premuto, l'algoritmo è in grado di generare risultati differenti (sempre che il testo di input abbia una lunghezza sufficiente per garantire ciò).

Bibliografia - Sitografia

[1] Bet On Math - Corso online tenuto dal Politecnico di Milano
in https://www.pok.polimi.it/courses/course-v1:Polimi+BOM101+2015_M11/info

[2] Lezione presso l'università di Harvard sulle Catene di Markov
in https://www.youtube.com/watch?v=8AJPs3gvNIY

[3] Markov Chains, Compact Lecture Notes and Exercises, ACC Coolen
in https://nms.kcl.ac.uk/ton.coolen/allnotes/MarkovChains.pdf

[4] Tempi discreti e tempi continui, Wikipedia
in https://it.wikipedia.org/wiki/Discreto_e_continuo

[5] Random Number Generation, Wikipedia
in https://en.wikipedia.org/wiki/Random_number_generation

[6] Pseudorandomness, Wikipedia
in https://en.wikipedia.org/wiki/Pseudorandomness

[7] Random.org
in https://www.random.org/randomness

[8] Confronto di PRNGs su sistemi operativi differenti, Bo Allen
in http://boallen.com/random-numbers.html

[9] Brian. W. Kernighan, Rob P, *Programmazione nella pratica (The practice of Programming)*

[10] How I Hacker News, Hacker News
in https://news.ycombinator.com/item?id=639976

[11] D.Alighieri, La Divina Commedia

[12] HTML
in http://www.w3schools.com/html

[13] CSS3
in http://www.w3schools.com/css/css3_intro.asp

[14] AngularJS
in https://angularjs.org

[15] Apache
in https://httpd.apache.org

[16] PHP 5
in http://php.net

[17] Unirest per PHP
in http://unirest.io/php.html

[18] Java
in https://www.java.com

[19] Mashape.com
in https://market.mashape.com

[20] “Learn to read and write Japanese kanji” API
in https://market.mashape.com

[21] Cloud9 - Online editor
in https://c9.io

[22] Git
in https://git-scm.com

[23] Material Design
in https://material.google.com

[24] Angular Material
in https://material.angularjs.org

[25] Material Icons
in https://design.google.com/icons

[26] La lingua giapponese, Wikipedia
in https://it.wikipedia.org/wiki/Lingua_giapponese

[27] I Kanji, Wikipedia
in https://it.wikipedia.org/wiki/Kanji