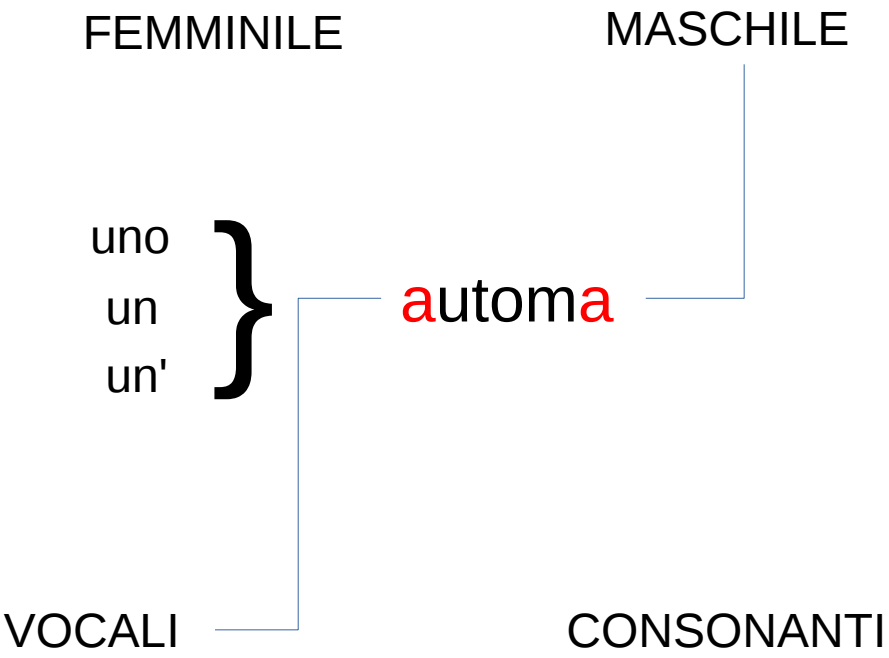
# E perché non un'automa?




*di Stefano Penge*

# Indice generale

# Oggetto

Questo documento descrive un ipotetica attività di "coding" da fare in classe. Il dominio è quello della lingua. L'obiettivo è costruire insieme un *automa linguistico*, cioè un programma in grado di simulare un parlante della lingua Italiana, in una situazione specifica molto semplice: premettere l'articolo ad un nome.

Si parte ricordando che la grammatica Italiana – a differenza di altre – prevede due tipi di articoli (determinativo e indeterminativo), che veicolano prevalentemente un'informazione sulla conoscenza da parte del parlante dell'insieme a cui appartiene l'individuo a cui si riferisce il sostantivo. Es.

Giovanni è venuto con il figlio *(ne ha uno solo)*

Giovanni è venuto con un figlio *(ne ha più di uno)*

A seconda però del genere del nome e della sua iniziale, si usano varianti diverse. I ragazzi (se sono di madre lingua Italiana) hanno una conoscenza almeno implicita che permette loro di riconoscere come errate le frasi:

* Giovanni è venuto con lo figlio

* Giovanni è venuto con la figlio

Qualche dubbio può venire per le frasi:

* Giovanni è venuto con un gnomo

* Giovanni è venuto con un altra figlia

e infatti è su questi casi che si concentrano la maggior parte degli eserciziari, digitali o meno.

Si inizia con una versione semplice, che è in grado di trattare solo alcuni casi; poi, dalla consapevolezza di questi limiti, si cerca di andare oltre, cambiando approccio. Per ogni passaggio, sono presentati degli esempi di codice sorgente e sono indicati alcuni suggerimenti di argomenti di discussione.

# Contesto

Perché è stato scelto un argomento così poco creativo, noioso, freddo, come quello della grammatica italiana, in particolare delle diverse forme degli articoli e il loro uso?

La prima cosa da notare è che il dominio non è STEM: non ci sono numeri, non ci sono grandezze fisiche. Questo per diversi motivi: prima di tutto, per offrire un esempio di esplorazione di campi diversi da quello scientifico, dove generalmente i docenti non hanno grandi competenze informatiche (e forse proprio per questo hanno bisogno di maggiori stimoli). Una mancanza di competenze che ha radici storiche profonde: la divisione delle arti, i curricoli scolastici, la separazione tra area umanistica e area scientifica; ma, a mio avviso, deriva anche da una generale incomprensione della natura dell'informatica, come tecnica che tratta numeri (in fondo "computer" significa "calcolatore", no?).

Informatica però significa "trattamento automatico dell'informazione", e informazione non significa quantità numeriche. La linguistica computazionale è una branca dell'informatica (o meglio, una disciplina all'incrocio di altre discipline) che ormai esiste da anni e ha una sua dignità che non deve essere più dimostrata.

Il fatto banale che nella lingua non ci sono numeri non significa che non si possa utilizzare un

computer in maniera interessante. Anzi: non solo si può, ma per certi versi è ancora più sensato. Se per lavorare in classe con la fisica e un computer o si usano sensori e attuatori, oppure si entra nel campo delle simulazioni (con i vantaggi e i limiti di questo approccio didattico), per lavorare con la lingua con un computer non serve niente altro. I computer sono tradizionalmente ben equipaggiati per trattare la lingua (soprattutto scritta, ma non solo). Sanno produrre stringhe di caratteri, frasi e persino testi veri, non solo *simulazioni di testi* (anche se nei diversi linguaggi di programmazione può essere più o meno semplice farlo) Il prodotto di un automa linguistico non è un disegno o un'animazione che va interpretata come schema di un fenomeno fisico: *è il fenomeno stesso.* Questa situazione, che è simile a quella della matematica, deriva dal fatto che la lingua può essere trattata come un insieme di simboli, che possono essere codificati, analizzati, combinati, letti e prodotti. Non significa naturalmente che *tutte* le competenze linguistiche possano essere apprese in questo modo; e infatti l'oggetto di questa proposta è all'interno del sottoinsieme della grammatica, intesa come insieme generale delle regole che governano la produzione di enunciati. Si potrebbe però fare un'operazione simile con le regole ad un livello più elevato, nel campo della logica o della narratologia, o della stilistica.

Perché gli articoli? Perché in Italiano sono abbastanza semplici (almeno a prima vista), ma non così semplici come, ad esempio, in Inglese. Perché si prestano a discutere di regole, di eccezioni, di ambiguità; permettono di andare a toccare la storia della lingua e le parentele tra lingue romanze; hanno aspetti fonetici e aspetti semantici; eccetera.

E a che età potrebbe essere proposta questa attività? L'oggetto è costituito da conoscenze di base, che fanno parte del programma della terza classe della scuola primaria. Tuttavia i concetti utilizzati nell'attività sono probabilmente troppo complessi per quell'età, almeno nella forma qui proposta. Si può quindi ipotizzare che si tratti di una ripresa di quelle conoscenze da parte di un gruppo di ragazzi più grandi, per esempio nella secondaria inferiore. Si può immaginare che lo scopo sia quello di costruire un programma che aiuti i bambini più piccoli, o di altre lingue madre; oppure può essere il punto di partenza per un confronto tra lingue diverse nel quadro delle attività di apprendimento della seconda lingua. Non lo so con certezza, e credo che spetti ad ogni docente trovare il momento ed il modo di introdurre un'attività nella sua classe, riducendola, modificandola, adattandola al contesto reale.

Non sono invece sicuro che il problema generale del rapporto tra grammatica e vocabolario, il tema della categorizzazione della parti del discorso e della sua importanza pratica, o le differenze tra le lingue, siano troppo difficili per poter essere affrontate anche nella scuola primaria. La grammatica probabilmente è considerata una "vecchia" maniera di affrontare la lingua in termini di adeguamento ad una norma; qui, al contrario, viene presentata come una maniera intelligente di affrontare tutte le possibili varianti di una istanza comunicativa. Categorizzare dei fenomeni e scoprire una regola significa poter trattare casi diversi da un unico punto di vista. E' una competenza che sta alla base di ogni attività scientifica, anche al di fuori del cosiddetto "pensiero computazionale".

In realtà l'oggetto è abbastanza indifferente, perché non si tratta di un'attività per imparare *cosa sono gli articoli e come si usano*. Costruire un automa in grado di assegnare la forma giusta di articolo ad un nome è soprattutto un modo di riflettere sulle competenze linguistiche apprese, di scomporle, di rappresentarle in modo da essere in grado di applicarle anche a casi particolari. E' anche, o soprattutto, un modo di acquisire consapevolezza della lingua.

## Modello didattico

Probabilmente i docenti di Lingua che leggeranno questo testo troveranno imprecisioni, errori, mancanze. Mea culpa: non sono un linguista, né un docente di lingua. Si può senz'altro correggere e migliorare, o ripensare da zero. In generale, quanto scritto rappresenta solo un esempio di una possibile modalità di utilizzo della programmazione come strumento di costruzione collettiva di conoscenze, da adattare (o ripensare) nei contesti specifici della classe.

L'esempio scelto è particolarmente semplice (almeno in apparenza, ma si vedrà che ci sono punti oscuri anche qui) per far emergere meglio una strategia didattica, che è basata non soltanto sulla programmazione, ma soprattutto sulla *discussione* che fa emergere ipotesi, sulla *sperimentazione* diretta, sul *miglioramento* incrementale, sulla proposta di *sviluppi*.

Non è un caso che l'esempio proposto non sia quello di un quiz o di un programma che "interroga" lo studente (come in tanti esempi che si possono trovare in rete[1]). Anzi: il modello è esplicitamente un altro, quello dell'insegnamento AL computer DA PARTE dello studente, o meglio da parte di un gruppo di studenti. Insegnamento di cosa? Di conoscenze e competenze che, almeno in parte, fanno parte del bagaglio di conoscenza implicita che può essere resa esplicita grazie proprio all'attività proposta, e in parte devono essere raccolte. Naturalmente è possibile che non tutti conoscano esattamente la lista delle condizioni di applicazione di una forma di articolo (per esempio, i ragazzi di lingua madre diversa dall'Italiano): ma fa parte – integrante – dell'attività proposta la ricerca di queste informazioni, attraverso l'intervista a utenti più esperti (ragazzi più grandi, genitori, docenti), la lettura di testo scolastico di grammatica Italiana, di un'enciclopedia[2] o di Wikipedia[3] – che rappresenta la sfida più difficile. Un'attività che non si svolge tutta davanti ad un computer, né è un'attività in solitaria: il gruppo è fondamentale per la discussione, per la ricerca, per la costruzione. Si possono anche organizzare due gruppi che si scambiano i ruoli: un gruppo scrive il programma, l'altro lo testa per vedere se si comporta "bene", cioè se risponde come farebbe un parlante Italiano.

Costruzione di conoscenza significa ricerca, non trasmissione. Il punto non è come trasmettere conoscenze agli studenti; il punto è come un gruppo di studenti arriva a costruire una conoscenza stabile, condivisibile, rappresentata in una forma pubblica, a partire da elementi sparsi, non strutturati, contraddittori o apparentemente tali. E' possibile che per alcuni studenti questa sia *anche* un'occasione per imparare da zero delle conoscenze di base, oltre che per costruire su quella base un modello funzionante; ma non è l'obiettivo primario, perché questa attività non si sostituisce ad una lezione di grammatica. Per intenderci: se si dovesse partire con lo *spiegare* che esistono articoli determinativi e indeterminativi, l'attività sarebbe troppo lunga e complessa e di dubbia efficacia. Qui si suppone semplicemente che il concetto sia almeno vagamente conosciuto, ma forse non utilizzato consapevolmente e al meglio; e che, viceversa, imparare a memoria un elenco di parole o saper rispondere ad una domanda non valga quanto la costruzione di una teoria esplicita su come si utilizza un pezzetto della lingua.

E il docente? Qui immaginiamo che il docente sia *fondamentale*. Il suo ruolo non è quello del trasmettitore, ma nemmeno quello del tutor che sovraintende alle attività di programmazione. Il docente è la guida, che apre la discussione, aiuta a vagliare le ipotesi, suggerisce parallelismi e approfondimenti. La sezione "Discussione" che segue ogni esempio di codice ha lo scopo di suggerire alcune piste che possono essere seguite dal docente in base al suo contesto. Il docente deve, naturalmente, avere un'ampia competenza sulla lingua (sulle lingue) e sulla glottodidattica,

1 Per esempio, http://www.baby-flash.com/lavagna/articoli_sbagliati1.swf oppure http://www.softwaredidatticofree.it/schedaapostrofo1.htm

2 http://www.treccani.it/enciclopedia/articoli-indeterminativi_(La-grammatica-italiana)/

3 https://it.wikipedia.org/wiki/Articolo_(linguistica)#Articoli_indeterminativi

ma anche sulla programmazione (vedi infra). Deve essere in grado di proporre delle tecniche alternative, di spiegarne i vantaggi e gli svantaggi. Da questo punto di vista, anche gli esempi di codice sorgente forniti non vanno presentati subito così come sono, ma costruiti, un pezzetto alla volta, discutendone la scrittura, provando versioni diverse, analizzando gli errori.

Una nota sul concetto di "problema", che spesso è presentato come il punto di partenza da cui discende la costruzione della soluzione algoritmica: qui non si parte da un *problema*, ma da un *progetto* di costruzione di qualcosa. Strada facendo, appaiono dei problemi (di lingua, o di linguaggio) che vanno affrontati, evitati o risolti. Alcuni si riveleranno alla fine problemi irrisolvibili con le risorse a disposizione, e anche questa è una scoperta importante.

## Competenze "computazionali"

Qualcuno dirà che queste discussioni e questi ragionamenti si sarebbe potuti fare tranquillamente *senza un computer*, magari usando carta e matita, oppure utilizzando un programmino già fatto da qualche docente volenteroso. La prima obiezione credo che non sia valida: senza uno strumento pratico per costruire un modello funzionante (un automa) non è possibile sperimentare davvero quel modello. Se si usa solo la lingua rappresentata staticamente alla lavagna (ad esempio con una tabella), ai ragazzi si deve chiedere di fidarsi del fatto che le regole insegnate funzionino davvero e coprano tutti i casi possibili. La lingua viene presentata come un proprietà di qualcun altro – mentre è patrimonio di tutti.

Ugualmente non mi convince la seconda obiezione (usare software didattico già pronto): una cosa è costruire un modello, un'altra è utilizzarlo senza sapere come è fatto "dentro". Non credo sia il caso di dilungarsi qui sul perché una modalità didattica basata sulla costruzione sia più efficace di una basata sull'uso. Si può, e si deve, discutere invece sul punto di partenza e di arrivo del processo di costruzione: non si può partire da zero e non si può pretendere di costruire un programma perfetto.

Detto questo, gli aspetti strettamente informatici (vedi sotto) in questa presentazione sono volutamente messi in secondo piano. Il motivo è che non sono il centro dell'attività: non si tratta di un'attività il cui scopo è quello di imparare, ad esempio, l'uso delle liste, ma un'attività *in cui* l'uso delle liste permette di costruire qualcosa di funzionante. Questa proposta non è mirata all'apprendimento del cosiddetto "pensiero computazionale" (qualsiasi cosa voglia dire), ma all'apprendimento della lingua.

Ma quindi queste competenze informatiche devono essere già possedute dagli studenti? Non è detto; anzi, l'attività proposta è anche un modo per mostrare come questi strumenti siano funzionali al problema affrontato e quindi per impararne l'uso in maniera significativa e motivante. Se gli studenti però li conoscono già, tanto meglio.

Un'ultima nota: l'ambiente scelto per l'esempio non è un ambiente visuale come Scratch o Snap!. Questa scelta è stata fatta, a dispetto della contraddizione apparente, per *facilitare* la lettura del codice sorgente, il suo commento, la sua modifica, il riuso di parti. In particolare, si tratta di Kojo[4], che è un ambiente didattico basato sul linguaggio Scala[5] e ricco di strumenti interessanti, tra cui un sottoinsieme di istruzioni logo-like per muovere un robot sullo schermo. Kojo permette di eseguire il codice in modalita "trace", ovvero mostrando in una finestra laterale il valore delle espressioni

4 http://www.kogics.net/sf:kojo
Un'introduzione in italiano: http://minimalprocedure.pragmas.org/writings/kojo-scala-appunti/kojo-scala-appunti.html
5 http://www.scala-lang.org/

man mano che viene calcolato, il che è molto utile per capire cosa succede.

E' possibile, se necessario, tradurre l'esempio in qualsiasi altro linguaggio che permette di costruire liste (o aggregazioni simili di caratteri). Tuttavia si è scelto un linguaggio funzionale e non imperativo perché questo modello si presta meglio a trattare l'argomento linguistico, perché è in generale più potente e, perché no, più elegante. Si tratta evidentemente anche di una sfida: i costrutti imperativi sono facili da interpretare come comandi impartiti ad un robot, e questa interpretazione è comprensibile anche per bambini più piccoli. Gli altri modelli di programmazione (funzionale, orientata agli oggetti, logica) richiedono un lavoro iniziale maggiore, che però varrebbe la pena di essere almeno valutato. Non è detto che partire da un dialogo tra "personaggi", o dalla costruzione di una dimostrazione, o dalla delegazione di un'elaborazione non possa essere un punto di partenza praticabile, se fatto con il lessico adeguato.

In appendice, comunque, presentiamo la traduzione del codice sorgente della versione 1 in due linguaggi diversi: Prolog (programmazione logica) e LibreLogo (programmazione funzionale). Le tre versioni presentano molte somiglianze e qualche differenza, alcune più evidenti (di tipo grammaticale e lessicale), alcune meno (di tipo strutturale). Anche la comparazione dei codice, pur senza approfondimenti, è un'attività interessante che può originare discussioni sul concetto di traduzione

Infine: tutto il codice sorgente,[6] per quel che vale, è rilasciato con licenza GPL 3.0. Significa che potete riusarlo, modificarlo, pubblicarlo. Credo che sia una buona abitudine dichiararlo.

## Concetti generali

- Grammatica e vocabolario (regole e fatti)
- Fonetica e pragmatica ("l'articolo esplicita l'opposizione pragmatica tra tema e rema")
- Automa linguistico
- Conoscenze (base di)
- Algoritmi
- Modello
- Categorizzazione
- Eccezioni
- ...

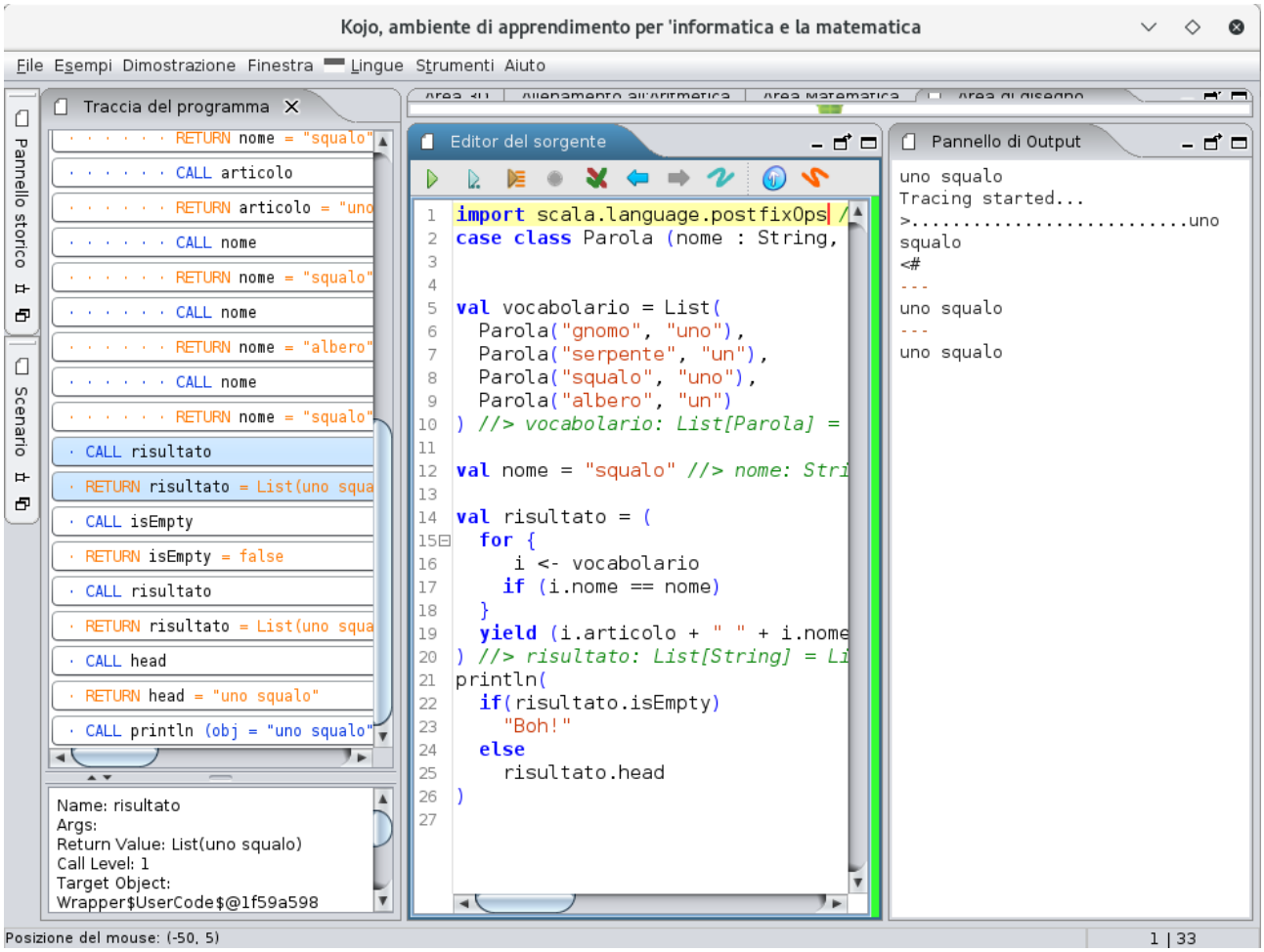
## Concetti "computazionali"

- Funzione e catene di funzioni
- Classe
- Oggetto
- Lista
- Stringhe e sottostringhe

6 Grazie a Massimo Ghisalberti che ha riscritto buona parte del codice in "bella". Gli errori rimasti sono opera mia.

- Mappatura di funzioni
- For
- Match
- Ricorsione

Elementi specifici del linguaggio Scala: *isEmpty, filter, contains, map, match, startsWith, endsWith, slice, toString, tail, head, get, yield, foldLeft, for, Some, None, Any, Option.*



## Versione 0: un approccio immediato

Si inizia con un sottoinsieme dell'obbiettivo: ci occupiamo solo dell'articolo *indeterminativo*, e solo di parole di genere *maschile*.

La strategia è semplice: l'automa riesce a mettere l'articolo davanti al nome perché *conosce esattamente ogni caso*. E' un'immagine di competenza piuttosto semplice: sa fare solo chi sa già. L'automa è immaginato come un deposito di fatti.

La sua conoscenza è costituita da una lista di coppie: il sostantivo e il suo articolo.

- gnomo, uno
- squalo, uno
- serpente, un
- albero, un

eccetera.

Le sue competenze sono costituite da una funzione che per ogni nome prova a cercare la versione corretta nella sua base di conoscenza, e se la trova, la restituisce.

Di seguito due possibili esempi di codice sorgente:

```
// Versione 0.1
// Copyright Stefano Penge 2017
// Released under GPL v.3.0

// vocabolario:
val vocabolario = Map (
  "gnomo"->"lo",
  "serpente"->"un",
  "squalo"->"uno",
  "albero"->"un"
)


val nome = "gnomo"

val risultato = vocabolario.get(nome)

risultato match {
  case Some(r) => println(r + " " + nome)
  case None => println("Boh!")
}
```

```
// Versione 0.2
// Copyright Stefano Penge 2017
// Released under GPL v.3.0

import scala.language.postfixOps
case class Parola (nome : String, articolo : String)


val vocabolario = List(
  Parola("gnomo", "uno"),
  Parola("serpente", "un"),
  Parola("squalo", "uno"),
  Parola("albero", "un")
)
```

```
val nome = "squalo"

val risultato = (
  for {
         i <- vocabolario
         if (i.nome == nome)
  }
 yield i.articolo + " " + i.nome
)
println(
  if(risultato.isEmpty)
         "Boh!"
  else
         risultato.head
)
```

### *Discussione*

Le due versioni del codice sono equivalenti, ma la seconda (leggermente più complessa) ha il pregio di poter essere facilmente estesa. Per esempio, per trattare anche l'articolo determinativo, come faremo più avanti, si potrebbe scrivere:

```
case class Parola (nome : String, articoloDet : String, articoloIndet : String)
val vocabolario = List(
  Parola("gnomo", "lo", "uno"),
  ...
```

L'automa risponde solo alle parole che conosce. Evidentemente questo approccio (basato su un vocabolario) non è sostenibile alla lunga, a meno di non avere a disposizione tutto il lessico italiano con associati gli articoli giusti. Esiste una cosa del genere[7]? Si può costruire un elenco completo, e se si , come? Che altre possibilità ci sono? Per esempio, come fa Google? Conosce tutte le risposte alle domande che gli faranno?

## Versione 1: un approccio basato sulla grammatica

Il passaggio successivo è utilizzare la grammatica, che qui viene vista come un modo efficace di gestire casi reali senza dover cablare tutti i fatti nel programma.

Probabilmente i ragazzi sono in grado di ricostruire quello che prevede la grammatica Italiana: si usa sempre "un", tranne che davanti a parole che cominciano con certe consonanti, o certe coppie di consonanti, o certe coppie di vocali. C'è una regola, e ci sono delle eccezioni (la "esse impura": che significa? e le semivocali?). Ci si può domandare la ragione di queste eccezioni. Si può discutere del perché una lingua abbia delle eccezioni, del fatto che in generale lingue diverse abbiano strutture più o meno regolari, fino a comparare le lingue naturali con i linguaggi artificiali, come la matematica o i linguaggi di programmazione, che non hanno eccezioni. Cosa sono, in fondo, le eccezioni? Quali cose hanno eccezioni?

Si può iniziare a scrivere una tabella come questa che segue:

7 In allegato 3 trovate una lista di circa 300 dei sostantivi più comuni

| ARTICOLO | INIZIALE |
|---|---|
| un | a,e,o,u; i seguite da consonante |
| uno | i seguita da altra vocale (semivocale) |
| un | consonante |
| uno | gn,pt,ps,sc,sf,sm,st, …. |

L'approccio cambia: ora la conoscenza è composta da:

- una lista di articoli
- una lista di vocali
- una lista di iniziali "particolari" (eccezioni)

Le competenze del programma devono aumentare. L'automa deve essere in grado di:

- capire se una parola comincia per vocale o consonante
- se comincia per vocale, capire se fa parte delle "eccezioni" per le vocali
- se invece comincia per consonante, capire se fa parte delle "eccezioni" per le consonanti
- assegnare l'articolo indeterminativo corretto

Ecco un esempio di codice sorgente:

```
// Versione 1
// Copyright Stefano Penge 2017
// Released under GPL v.3.0

class Articoli  {
val artInDetList = List("un","uno")
val consonantiNormaliList = List ("b","c","d","f","g","h","l","m","n","p","q","r","s","t","v")
val consonantiStraneList = List
("z","gn","pn","ps","sb","sc","sf","sg","sl","sm","sn","sp","sq","st","sv")
val vocNormaliList = List("a","e","i","o","u")
val vocStraneList = List("ia","ie","io","iu")


def iniziaPerVocale (nome: String) : Boolean = {
    !vocNormaliList.filter(
      nome.startsWith(_)
    ).isEmpty
}

def iniziaPerVocaleStrana (nome: String) : Boolean = {
    !vocStraneList.filter(
      nome.startsWith(_)
    ).isEmpty
}

def iniziaPerConsonanteStrana (nome: String) : Boolean = {
    !consonantiStraneList.filter(
      nome.startsWith(_)
```

```
    ).isEmpty
}

def iniziaPerConsonanteNormale (nome: String) : Boolean = {
    !consonantiNormaliList.filter(
      nome.startsWith(_)
    ).isEmpty
}



def trovaArticolo (nome: String) : Any = {
  // trova l'articolo indeterminativo giusto

  val articoli =  artInDetList


  if (iniziaPerVocaleStrana(nome)) // è una vocale strana
    return articoli.last

  if (iniziaPerVocale(nome)) // è una vocale normale
    return articoli.head

  if (iniziaPerConsonanteStrana(nome)) //  è una consonante strana
    return articoli.last

  if (iniziaPerConsonanteNormale(nome)) //  è una consonante normale
    return articoli.head

} // end trovaArticolo

} // end class Articoli


// main
val automa_articoli = new Articoli

val nome = "squalo"
println(automa_articoli.trovaArticolo(nome) + " " + nome)
```

## *Discussione*

La versione del codice presentato utilizza delle forme nuove. Per i dati è stata utilizzata la lista, che è un tipo di dati universale.

Perché è così importante l'iniziale di una parola?  Quanto sono frequenti le parole che iniziano per vocale?

Si può andare avanti? Certo: all'automa, in questa versione, manca la gestione degli articoli determinativi. Ma l'approccio dovrebbe essere lo stesso. Gli articoli determinativi funzionano come quelli indeterminativi?

## Versione 2: gestire anche gli articoli determinativi

Per verificarlo, si può costruire uno schema simile a quello precedente, riassumendo le regole che valgono per gli articoli determinativi maschili:

| ARTICOLO | INIZIALE |
|---|---|
| lo | a,e,o,u; i seguite da consonante |
| lo | i seguita da altra vocale (semivocale) |
| il | consonante |
| lo | gn,pt,ps,sc,sf,sm,st, …. |

Un esempio di codice sorgente:

```
// Versione 2: articoli determinativi
// Copyright Stefano Penge 2017
// Released under GPL v.3.0

class Articoli  {
val artInDetList = List("un","uno")
val artDetList = List ("il","lo")
val artSconList = List ()
val consonantiNormaliList = List ("b","c","d","f","g","h","l","m","n","p","q","r","s","t","v")
val consonantiStraneList = List
("z","gn","pn","ps","sb","sc","sf","sg","sl","sm","sn","sp","sq","st","sv")
val vocNormaliList = List("a","e","i","o","u")
val vocStraneList = List("ia","ie","io","iu")


def iniziaPer (iniziali: List[String], nome : String) : Boolean = {
    iniziali.filter(nome.startsWith(_)) match {
     case Nil => false;
     case _ => true
   }
}


def trovaTipoArticolo (articolo: String) : List[String] = {
    // trova il tipo e restituisce la lista di articoli adeguata

   if (artInDetList.contains(articolo))
     return artInDetList

   if (artDetList.contains(articolo))
     return artDetList

   return artSconList

}

def trovaArticolo (nome: String) : Any = {
  // trova l'articolo giusto
```

```
  val articoli =  trovaTipoArticolo(articolo)
  if (articoli.isEmpty)
    return "Boh"

  if ((iniziaPer(vocStraneList, nome)) || (iniziaPer(consonantiStraneList, nome)))
    return articoli.last

  if ((iniziaPer(vocNormaliList, nome)) || (iniziaPer(consonantiNormaliList, nome)))
    return articoli.head

}


} // end class Articoli


// main
val automa_articoli = new Articoli
val articolo = "il"
val nome = "gnomo"
println(automa_articoli.trovaArticolo(nome) + " " + nome)
```

## *Discussione*

Può valer la pena andare a scoprire come sono nati gli articoli determinativi in Italiano (da "ille", "illa" e "illud").

Il caso dell'iniziale vocalica presenta un problema di allitterazione (* lo albero) che per ora si può rimandare, ma andrà risolto in seguito. Un problema simile si trova in altre lingue "parenti" dell'Italiano: Francese, Catalano... ma non in Spagnolo. Perché la lingua non ama le allitterazioni? E in poesia, invece?

Il codice sorgente è leggermente più corto: invece di quattro funzioni diverse per scoprire le iniziali, se ne è usata una sola:

```
       def iniziaPer (iniziali: List[String], nome : String) : Boolean
```

Inoltre, per trovare l'articolo giusto, i quattro casi si sono ridotti a due (anzi, a tre, perché può succedere che l'articolo proposto non sia riconosciuto dall'automa, e allora...). Durante l'esecuzione, l'automa potrebbe scrivere anche dei risultati parziali:

```
        println(articolo+": Tipo indeterminativo")
```

Questo non è essenziale, ma è uno dei modi per aiutare la verifica di quello che sta facendo e per trovare più facilmente errori. La ricerca del bug in un programma è un processo difficile: gli errori di ortografia e sintassi si vedono subito (i linguaggi visuali non li permettono proprio), ma quelli di logica sono più sottili. Eppure rileggendo per la millesima volta un codice alla ricerca di un errore si imparano molte cose. A proposito, che differenza c'è tra un errore in un programma e un errore in un testo scritto in un linguaggio naturale? Anche lì ci sono errori di ortografia, di sintassi, di struttura. Quali sono quelli più gravi? Che significa, per un errore, essere "grave"?

A questo punto ci si sarà accorti di una mancanza fondamentale: manca la gestione delle parole femminili. Ma come si fa a capire se un sostantivo è femminile o maschile? Quanti sostantivi Italiani terminano per "a" o "o"? E' una caratteristica dell'Italiano? Perché?

## Versione 3: genere ed elisione

Bisogna aggiungere alle conoscenze l'elenco degli articoli femminili, determinativi e indeterminativi.

In più, va aggiunto l'elenco delle vocali finali che sono indizio di genere: "a","o".

La tabella qui è più semplice perché c'è una sola forma, "la" e "una", che viene usata sempre, tranne i casi in cui il sostantivo inizia per vocale, in cui c'è l'elisione.

L'automa deve essere in grado di utilizzare la forma elisa "un' " per i sostantivi femminili che cominciano per vocale (ma non semivocale, come "iella").

Già che ci siamo, possiamo anche sostituire l'ultima vocale anche in "lo" e "la" se il sostantivo comincia per vocale. Il riuso di uno schema che funziona è un buon trucco per evitare di ricominciare sempre da zero.

Un esempio di codice sorgente:

```
// Versione 3: genere
// Copyright Massimo Ghisalberti 2017
// Released under GPL v.3.0


object Articoli {
  val Determinativo = "determinativo"
  val Indeterminativo = "indeterminativo"
  val Maschile = "maschile"
  val Femminile = "femminile"
  val Elisione = "elisione"
  val Boh = "Boh!"
}

class Articoli {

  import Articoli._

  val articoli = Map(
    Indeterminativo -> Map(
      Maschile -> List("un", "uno"),
      Femminile -> List("una"),
      Elisione -> List("un'")
    ),
    Determinativo -> Map(
      Maschile -> List("il", "lo"),
      Femminile -> List("la"),
      Elisione -> List("l'")
    )
  )
  val vocaliPerGenere = Map(
    Maschile -> List('i', 'o', 'u'),
    Femminile -> List('a', 'e')
  )

  val iniziali = List("x", "y", "z", "gn", "pn", "ps", "sb", "sc", "sf", "sg", "sl", "sm", "sn",
"sp", "sq", "st", "sv")

  def vocali: List[Char] = vocaliPerGenere.foldLeft(List[Char]())((acc, l) => acc ++ l._2)

  def trovaArticolo(articolo: String): Option[(String, String, String)] = {
    val ret = for {
      tipo <- articoli.keys
```

```
      generi = articoli(tipo)
      genere <- generi.keys
      if generi(genere).contains(articolo)
    } yield (tipo, genere, articolo)
    if (ret.isEmpty) None else Some(ret.head)
  }

  def iniziaPerVocale(nome: String): Boolean = vocali.contains(nome.head)

  def iniziaPerCoppia(nome: String): Boolean = {
    val coppia = nome.slice(0, 2)
    iniziali.contains(coppia.head.toString) ||
    iniziali.contains(coppia) ||
    (vocali.contains(coppia.head.toString) && vocali.contains(coppia.last.toString)) ||
    (iniziali.contains(coppia.head.toString) && !vocali.contains(coppia.last.toString))
  }

  def genereMaschile(nome: String): Boolean = {
    val l = nome.toList
    l.last == 'o' || l.last == 'i' || l.last == 'e'
  }

  def generefemminile(nome: String): Boolean = {
    val l = nome.toList
    l.last == 'a' || l.last == 'e'
  }

  def genere(nome: String): String = if (genereMaschile(nome)) Maschile else Femminile

  def mettiArticolo(nome: String, tipo: String): String = {
    val gen = genere(nome)
    val articolo = tipo match {
      case Determinativo => {
        if (iniziaPerVocale(nome)) articoli(tipo)(gen).last
        else if (iniziaPerCoppia(nome)) articoli(tipo)(gen).last
        else if (!iniziaPerVocale(nome)) articoli(tipo)(gen).head
        else Boh
      }
      case Indeterminativo =>
        if (iniziaPerVocale(nome)) articoli(tipo)(gen).head
        else if (iniziaPerCoppia(nome)) articoli(tipo)(gen).last
        else if (!iniziaPerVocale(nome)) articoli(tipo)(gen).last
        else Boh
      case _ => Boh
    }
    s"${articolo} ${nome}"
  }
}

// main
val automa_articoli = new Articoli


List("gnoma", "squalo", "serpente", "albero", "ala", "pane", "albatro").foreach { nome =>
  println(automa_articoli.mettiArticolo(nome, Articoli.Indeterminativo))
}
List("gnoma", "squalo", "serpente", "albero", "ala", "pane", "albatro").foreach { nome =>
  println(automa_articoli.mettiArticolo(nome, Articoli.Determinativo))
}
```

## *Discussione*

Il codice poteva essere scritto in molti modi diversi. In questo caso sono state usate delle caratteristiche più tipiche dei linguaggi funzionali. L'uso di un costrutto anziché un altro è questione di competenza, ma anche di stile. Non è una scelta irrilevante: un codice più compatto è più facile da gestire, ma uno lungo e ripetitivo potrebbe essere più leggibile da parte di chi non l'ha scritto. Quanto è importante la leggibilità? Che impatto ha sulla "vita" di un testo, sul suo passare da un autore ad un altro? E cosa implica la modifica di un testo scritto da altri? E' la stessa cosa se parliamo di romanzi, articoli di Wikipedia o di programmi?

Alla fine del codice sorgente sono state aggiunte due espressioni che permettono di testare l'automa con più parole in una sola volta. Mostrano una caratteristica potente dei linguaggi funzionali come Scala: quella di trattare le funzioni come dati. E', tra l'altro, una delle differenze tra Snap! e Scratch.

L'automa a questo punto sembra sufficientemente robusto, cioè se la cava abbastanza bene nella maggior parte dei casi – ma ci si può divertire a testarlo con parole inesistenti, come "iisterico", "gtondo", "szero".

C'è però un problema che sembra irrisolvibile: alcuni sostantivi italiani terminano al singolare per "e" o per consonante, e non è possibile da questo dedurre il genere (ovvero ci sono casi di genere maschile come "pane" e casi di genere femminile come "fame"). Qui l'analisi della singola parola non è sufficiente. Si apre di nuovo la discussione tra approccio basato su vocabolario e approccio basato su grammatica. Si può usare un approccio misto? Può aiutare la forma del verbo associato (ad esempio, se una parola termina per "e" e c'è un verbo al plurale, allora è femminile)?

Anche una volta raggiunto l'obiettivo iniziale, si può andare ancora avanti. Si può applicare ad altre lingue parenti dell'Italiano. Anche all'Inglese? E alle lingue che hanno tre generi? E se invece degli articoli provassimo a trattare gli aggettivi dimostrativi (questo, quello)?

Se volessimo riusarne delle parti, non sarebbe meglio riorganizzarlo in moduli separati riusabili ?

Si può pensare di tradurre il programma in altri linguaggi  di programmazione (tutti? I linguaggi sono equivalenti, come le lingue?). Cosa cambierebbe? Che requisiti dovrebbe avere il nuovo linguaggio per tradurre facilmente il codice già scritto? Dando un'occhiata veloce ai codici sorgenti forniti in altri due linguaggi (Prolog e LibreLogo), cosa si vede?

# Allegato 1: versione LibreLogo

https://help.libreoffice.org/Writer/LibreLogo_Toolbar
http://librelogo.org/en/
*http://iamarf.ch/unifi/Piccolo-manuale-LibreLogo.pdf*

```
;Nota: per eseguire il codice è sufficiente incollarlo dentro una pagina di
;Libre Office Writer in cui sia stata installata la ToolBar LibreLogo.
;Viene mostrata una finestra di dialogo che chiede di scrivere una parola.

TO vai
HIDETURTLE
:parola = INPUT “Parola?”
PRINT  articolo :parola
END

TO articolo :parola
      OUTPUT trova_articolo (trova_tipo :parola)
END


TO trova_tipo :parola
IF (inizia_vocale_strana :parola) [ OUTPUT “eccezione” ] ; uno
IF (inizia_vocale :parola) [ OUTPUT “base” ] ; un
IF (inizia_consonante_strana :parola) [ OUTPUT “eccezione” ] ; uno
IF (inizia_consonante :parola) [ OUTPUT “base” ]
OUTPUT "BOH"
END

TO  trova_articolo :tipo
IF (:tipo == “base”) [
      OUTPUT “un”
][
      IF (:tipo == “eccezione”)[
            OUTPUT “uno”
      ][
            OUTPUT :tipo
      ]
]
END

TO inizia_vocale :parola
      :vocali = [“a”,“e”,“i”,“o”,“u”]
      :iniziale = trova_iniziale(:parola,0)
      FOR :lettera IN [“a”,“e”,“i”,“o”,“u”] [
            IF  (:iniziale == :lettera)[ OUTPUT TRUE ]
      ]
      OUTPUT FALSE
END

TO inizia_vocale_strana :parola
      :iniziale = trova_iniziale(:parola,0)
      :semivocale = “i”
      IF  (:iniziale == :semivocale)[
            :vocali = [“a”,“e”,“o”,“u”]
            :seconda = trova_iniziale  (:parola,2)
            FOR :lettera IN :vocali [
```

```
                  IF  (:seconda == :lettera)[ OUTPUT TRUE ]
            ]
            OUTPUT FALSE
      ]
      OUTPUT FALSE
END

TO inizia_consonante :parola
      :consonanti =
[“b”,“c”,“d”,“f”,“g”,“h”,“l”,“m”,“n”,“p”,“q”,“r”,“s”,“t”,“v”]
      :iniziale = trova_iniziale(:parola,0)
      FOR :lettera IN  :consonanti [
            IF  (:iniziale == :lettera)[ OUTPUT TRUE ]
      ]
      OUTPUT FALSE
END

TO inizia_consonante_strana :parola
      :iniziale = trova_iniziale (:parola,0)
      IF  (:iniziale == “z”)[ OUTPUT TRUE ][
            :iniziali = trova_iniziali (:parola, 2)
            :consonanti = [“gn”,“pn”,“sc”,“sf”,“sq”,“st”]
            FOR :coppia IN :consonanti [
                  IF  (:coppia == :iniziali)[ OUTPUT TRUE ]
            ]
      ]
      OUTPUT FALSE
END

TO trova_iniziale :parola :indice
      p = :parola
      OUTPUT p[:indice]
END

TO trova_iniziali :parola :quante
      p = :parola
      q = :quante
      OUTPUT p[0:q]
END

vai
```

# Allegato 2: versione Prolog

https://it.wikipedia.org/wiki/SWI-Prolog
http://www.swi-prolog.org/
http://www.lynxlab.com/staff/steve/public/cma/prolog_intro_tutorial.pdf

```
/* Nota: per eseguire il codice occorre prima salvarlo come file di testo (ad
esempio come articoli.pl), poi lanciare l'interprete Prolog (es. swipl o
gprolog), infine caricare il sorgente (scrivendo: consult(articoli).) A questo
punto Prolog è pronto per rispondere a domande del tipo:
      metti_articolo(gnomo,Articolo).
ovvero: dimmi l'articolo giusto da mettere davanti a "gnomo", oppure:
      metti_articolo(zuzzurellone,un).
Ovvero: controlla se è giusto mettere "un" davanti a "zuzzurellone".
Può essere interessante attivare il tracing (scrivendo: trace.) per vedere come
l'interprete Prolog cerca di trovare la risposta.
*/

articolo_indeterminativo(base,un).
articolo_indeterminativo(eccezione,uno).
vocali(vocali_normali,[a,e,i,o,u]).
vocali(semivocali,[i]).
consonanti(consonanti_normali,[b,c,d,f,g,h,l,m,n,p,q,r,s,t,v]).
consonanti(consonanti_strane,[z,x]).
consonanti(coppie,[gn,pn,ps,sb,sc,sf,sg,sl,sm,sn,sp,sq,st,sv]).


inizia_per_vocale_strana(Parola) :-
/* inizia con una i seguita da altra vocale? */
      atom_chars(Parola,Lista_caratteri),
      [Iniziale,Seconda|_] = Lista_caratteri,
      vocali(semivocali,Lista_semivocali),
      member(Iniziale,Lista_semivocali),
      vocali(vocali_normali,Lista_vocali),
      member(Seconda,Lista_vocali).

inizia_per_vocale_normale(Parola) :-
/* inizia con una vocale seguita da consonante? */
      atom_chars(Parola,Lista_caratteri),
      [Iniziale,Seconda|_] = Lista_caratteri,
      vocali(vocali_normali,Lista_vocali),
      member(Iniziale,Lista_vocali),
      consonanti(consonanti_normali,Lista_consonanti),
      member(Seconda,Lista_consonanti).

inizia_per_consonante_strana(Parola) :-
/* inizia con una z o x ? */
      atom_chars(Parola,Lista_caratteri),
      [Iniziale|_] = Lista_caratteri,
      consonanti(consonanti_strane,Lista_consonanti),
      member(Iniziale,Lista_consonanti).

inizia_per_consonante_strana(Parola) :-
/* inizia con esse impura, gn etc ? */
      atom_chars(Parola,Lista_caratteri),
      [Iniziale,Seconda|_] = Lista_caratteri,
      atom_chars(Coppia,[Iniziale,Seconda]),
```

```
      consonanti(coppie,Lista_coppie),
      member(Coppia,Lista_coppie).


inizia_per_consonante_normale(Parola) :-
/* inizia con consonante seguita da vocale ? */
      atom_chars(Parola,Lista_caratteri),
      [Iniziale|Seconda] = Lista_caratteri,
      consonanti(consonanti_normali,Lista_consonanti),
      member(Iniziale,Lista_consonanti),
      vocali(vocali_normali,Lista_vocali),
      member(Seconda,Lista_vocali).

articolo(Parola,Articolo):-
      inizia_per_vocale_strana(Parola),
      articolo_indeterminativo(eccezione,Articolo).

articolo(Parola,Articolo):-
      inizia_per_vocale_normale(Parola),
      articolo_indeterminativo(base,Articolo).

articolo(Parola,Articolo):-
      inizia_per_consonante_strana(Parola),
      articolo_indeterminativo(eccezione,Articolo).

articolo(Parola,Articolo):-
      inizia_per_consonante_normale(Parola),
      articolo_indeterminativo(base,Articolo).

metti_articolo(Parola,Articolo) :-
      articolo(Parola,Articolo),
      write(Articolo),
      write(' '),
      write(Parola),
      nl.
```

# Allegato 3: una lista dei più comuni sostantivi Italiani

(da http://www.scudit.net/md333nomi_2.htm)

| *N* | ART DET NOME | GENERE | ART DET PLUR NOME |
|---|---|---|---|
| ***1*** | L' abitante | **Sm e Sf** | Gli e Le abitanti |
| ***2*** | L' acqua | **Sm** | Le (acque) |
| ***3*** | L' aereo | **Sm** | Gli aerei |
| ***4*** | L' aeroporto | **Sm** | Gli aeroporti |
| ***5*** | (L') agosto | **Sm** | - - |
| ***6*** | L' aiuto | **Sm** | Gli aiuti |
| ***7*** | L' albergo | **Sm** | Gli alberghi |
| ***8*** | L' albero | **Sm** | Gli alberi |
| ***9*** | L' amica/amico | **Sf/Sm** | Le/Gli amiche/amici |
| ***10*** | L' amore | **Sm** | Gli amori |
| ***11*** | L' andata | **Sf** | Le (andate) |
| ***12*** | L' animale | **Sm** | Gli animali |
| ***13*** | L' anno | **Sm** | Gli anni |
| ***14*** | L' appartamento | **Sm** | Gli appartamenti |
| ***15*** | (L') aprile | **Sm** | - - |
| ***16*** | L' arrivo | **Sm** | Gli arrivi |
| ***17*** | L' arte | **Sf** | Le arti |
| ***18*** | L' artista | **Sm e Sf** | Gli e Le artisti e artiste |
| ***19*** | L' attenzione | **Sf** | Le attenzioni |
| ***20*** | L' auto | **Sf** | Le auto |
| ***21*** | L' autobus | **Sm** | Gli autobus |
| ***22*** | L' autunno | **Sm** | Gli autunni |
| ***23*** | Il bacio | **Sm** | I baci |
| ***24*** | Il bagno | **Sm** | I bagni |
| ***25*** | La/Il bambina/bambino | **Sf/Sm** | Le/I bambine/bambini |
| ***26*** | La banca | **Sf** | Le banche |
| ***27*** | Il bar | **Sm** | I bar |
| ***28*** | Il e La barista | **Sm e Sf** | I e Le baristi e bariste |
| ***29*** | Il bicchiere | **Sm** | I bicchieri |
| ***30*** | La bicicletta | **Sf** | Le biciclette |
| ***31*** | Il biglietto | **Sm** | I biglietti |
| ***32*** | Il binario | **Sm** | I binari |
| ***33*** | La birra | **Sf** | Le birre |
| ***34*** | La bistecca | **Sf** | Le bistecche |
| ***35*** | La bocca | **Sf** | Le bocche |
| ***36*** | La borsa | **Sf** | Le borse |
| ***37*** | La bottiglia | **Sf** | Le bottiglie |
| ***38*** | La bruschetta | **Sf** | Le bruschette |
| ***39*** | Il caffè | **Sm** | I caffè |
| ***40*** | Il caffelatte | **Sm** | I caffellatte |

| | | | |
|---|---|---|---|
| ***41*** | Il calendario | **Sm** | I calendari |
| ***42*** | La camera | **Sf** | Le camere |
| ***43*** | La/Il cameriera/cameriere | **Sf/Sm** | Le/I cameriere/camerieri |
| ***44*** | La camicia | **Sf** | Le camicie |
| ***45*** | La campagna | **Sf** | Le campagne |
| ***46*** | Il cane | **Sm** | I cani |
| ***47*** | Il e La cantante | **Sm e Sf** | I e Le cantanti |
| ***48*** | La canzone | **Sf** | Le canzoni |
| ***49*** | Il capello | **Sm** | I capelli |
| ***50*** | La capitale | **Sf** | Le capitali |
| ***51*** | (Il) capodanno | **Sm** | I capodanno |
| ***52*** | Il cappuccino | **Sm** | I cappuccini |
| ***53*** | Il carabiniere | **Sm** | I carabinieri |
| ***54*** | Il carattere | **Sm** | I caratteri |
| ***55*** | La carne | **Sf** | Le (carni) |
| ***56*** | (Il) carnevale | **Sm** | I (carnevale) |
| ***57*** | La carta di credito | **Sf** | Le carte di credito |
| ***58*** | La casa | **Sf** | Le case |
| ***59*** | La casalinga | **Sf** | Le casalinghe |
| ***60*** | Il casino | **Sm** | I casini |
| ***61*** | La cassa | **Sf** | Le casse |
| ***62*** | La cena | **Sf** | Le cene |
| ***63*** | Il centro | **Sm** | I centri |
| ***64*** | La chiave | **Sf** | Le chiavi |
| ***65*** | La chiesa | **Sf** | Le chiese |
| ***66*** | Il chilo | **Sm** | I chili |
| ***67*** | Il chilometro | **Sm** | I chilometri |
| ***68*** | Il cielo | **Sm** | I (cieli) |
| ***69*** | Il cinema | **Sm** | I cinema |
| ***70*** | La/Il cioccolata/cioccolato | **Sf/Sm** | Le/I cioccolate/cioccolati |
| ***71*** | La città | **Sf** | Le città |
| ***72*** | Il cognome | **Sm** | I cognomi |
| ***73*** | (La) colazione | **Sf** | Le colazioni |
| ***74*** | Il colore | **Sm** | I colori |
| ***75*** | Il coltello | **Sm** | I coltelli |
| ***76*** | Il compleanno | **Sm** | I compleanni |
| ***77*** | Il cornetto | **Sm** | I cornetti |
| ***78*** | Il costo | **Sm** | I costi |
| ***79*** | La cucina | **Sf** | Le cucine |
| ***80*** | La/Il cugina/cugino | **Sf/Sm** | Le/I cugini |
| ***81*** | La cultura | **Sf** | Le culture |
| ***82*** | La destra | **Sf** | Le (destre) |
| ***83*** | Il dialetto | **Sm** | I dialetti |
| ***84*** | (Il) dicembre | **Sm** | - - |

| | | | |
|---|---|---|---|
| ***85*** | Il/La direttore/direttrice | **Sm/Sf** | I/Le direttori/direttrici |
| ***86*** | La discoteca | **Sf** | Le discoteche |
| ***87*** | La doccia | **Sf** | Le docce |
| ***88*** | La domanda | **Sf** | Le domande |
| ***89*** | La domenica | **Sf** | Le domeniche |
| ***90*** | La donna | **Sf** | Le donne |
| ***91*** | Il/La dottore/dottoressa | **Sm/Sf** | Il/La dottori/dottoresse |
| ***92*** | La droga | **Sf** | Le droghe |
| ***93*** | L' entrata | **Sf** | Le entrate |
| ***94*** | L' erba | **Sf** | Le (erbe) |
| ***95*** | L' errore | **Sm** | Gli errori |
| ***96*** | L' esempio | **Sm** | Gli esempi |
| ***97*** | L' est | **Sm** | - - |
| ***98*** | L' estate | **Sf** | Le estati |
| ***99*** | L' euro | **Sm** | Gli euro |
| ***100*** | La fame | **Sf** | - - |
| ***101*** | La famiglia | **Sf** | Le famiglie |
| ***102*** | La fantasia | **Sf** | Le fantasie |
| ***103*** | (Il) febbraio | **Sm** | - - |
| ***104*** | La femmina | **Sf** | Le femmine |
| ***105*** | - - | **Sf** | Le ferie |
| ***106*** | La fermata | **Sf** | Le fermate |
| ***107*** | (Il) Ferragosto | **Sm** | I (Ferragosto) |
| ***108*** | La festa | **Sf** | Le feste |
| ***109*** | La/Il fidanzata/fidanzato | **Sf/Sm** | Le/I fidanzate/fidanzati |
| ***110*** | La/Il figlia/figlio | **Sf/Sm** | Le/I figlie/figli |
| ***111*** | Il film | **Sm** | I film |
| ***112*** | La finestra | **Sf** | Le finestre |
| ***113*** | Il fiore | **Sm** | I fiori |
| ***114*** | La firma | **Sf** | Le firme |
| ***115*** | Il fiume | **Sm** | I fiumi |
| ***116*** | Il formaggio | **Sm** | I formaggi |
| ***117*** | La fortuna | **Sf** | Le (fortune) |
| ***118*** | La foto(grafia) | **Sf** | Le foto(grafie) |
| ***119*** | Il fratello | **Sm** | I fratelli |
| ***120*** | La frutta | **Sf** | - - |
| ***121*** | Il fumo | **Sm** | I (fumi) |
| ***122*** | Il fuoco | **Sm** | I fuochi |
| ***123*** | Il futuro | **Sm** | - - |
| ***124*** | Il gatto | **Sm** | I gatti |
| ***125*** | Il gelato | **Sm** | I gelati |
| ***126*** | Il genitore | **Sm** | I genitori |
| ***127*** | (Il) gennaio | **Sm** | - - |
| ***128*** | La gente | **Sf** | Le (genti) |
| ***129*** | La giacca | **Sf** | Le giacche |
| ***130*** | Il giardino | **Sm** | I giardini |

| | | | |
|---|---|---|---|
| ***131*** | Il gioco | **Sm** | I giochi |
| ***132*** | Il giornale | **Sm** | I giornali |
| ***133*** | Il e La giornalista | **Sm e Sf** | I e Le giornalisti e giornaliste |
| ***134*** | Il giorno | **Sm** | I giorni |
| ***135*** | Il giovedì | **Sm** | I giovedì |
| ***136*** | (Il) giugno | **Sm** | - - |
| ***137*** | Il gruppo | **Sm** | I gruppi |
| ***138*** | L' idea | **Sf** | Le idee |
| ***139*** | L' immigrato | **Sm** | Gli immigrati |
| ***140*** | L' immigrazione | **Sf** | - - |
| ***141*** | L' indirizzo | **Sm** | Gli indirizzi |
| ***142*** | L' informazione | **Sf** | Le informazioni |
| ***143*** | L' insalata | **Sf** | Le insalate |
| ***144*** | L' insegnante | **Sm e Sf** | Gli e Le insegnanti |
| ***145*** | L' inverno | **Sm** | Gli inverni |
| ***146*** | L' isola | **Sf** | Le isole |
| ***147*** | L' istituto | **Sm** | Gli istituti |
| ***148*** | Il lago | **Sm** | I laghi |
| ***149*** | Il latte | **Sm** | - - |
| ***150*** | Il/La lavoratore/lavoratrice | **Sm/Sf** | Il/La lavoratore/lavoratrici |
| ***151*** | Il lavoro | **Sm** | I lavori |
| ***152*** | Il legno | **Sm** | (I) (legni) |
| ***153*** | Il letto | **Sm** | I letti |
| ***154*** | La libertà | **Sf** | Le libertà |
| ***155*** | Il libro | **Sm** | I libri |
| ***156*** | La lingua | **Sf** | Le lingue |
| ***157*** | (Il) luglio | **Sm** | - - |
| ***158*** | La luna | **Sf** | Le lune |
| ***159*** | Il lunedì | **Sm** | I lunedì |
| ***160*** | La macchina | **Sf** | Le macchine |
| ***161*** | La madre | **Sf** | Le madri |
| ***162*** | La mafia | **Sf** | Le mafie |
| ***163*** | Il mafioso | **Sm** | I mafiosi |
| ***164*** | (Il) maggio | **Sm** | - - |
| ***165*** | Il malato | **Sm** | I malati |
| ***166*** | Il male | **Sm** | I mali |
| ***167*** | La mamma | **Sf** | Le mamme |
| ***168*** | La mano | **Sf** | Le mani |
| ***169*** | Il mare | **Sm** | I mari |
| ***170*** | Il marito | **Sm** | I mariti |
| ***171*** | Il martedì | **Sm** | I martedì |
| ***172*** | (Il) marzo | **Sm** | - - |
| ***173*** | La mattina | **Sf** | Le mattine |
| ***174*** | La medicina | **Sf** | Le medicine |
| ***175*** | Il medico | **Sm** | I medici |

| | | | |
|---|---|---|---|
| ***176*** | Il mercoledì | **Sm** | I mercoledì |
| ***177*** | Il mese | **Sm** | I mesi |
| ***178*** | Il metallo | **Sm** | I metalli |
| ***179*** | La metro(politana) | **Sf** | Le metro(politane) |
| ***180*** | La mezzanotte | **Sf** | - - |
| ***181*** | Il mezzogiorno | **Sm** | - - |
| ***182*** | La minestra | **Sf** | Le minestre |
| ***183*** | Il minestrone | **Sm** | I minestroni |
| ***184*** | Il minuto | **Sm** | I minuti |
| ***185*** | La moda | **Sf** | Le mode |
| ***186*** | La moglie | **Sf** | Le mogli |
| ***187*** | Il momento | **Sm** | I momenti |
| ***188*** | Il mondo | **Sm** | I mondi |
| ***189*** | La montagna | **Sf** | Le montagne |
| ***190*** | Il monumento | **Sm** | I monumenti |
| ***191*** | Il muro | **Sm** | I muri |
| ***192*** | Il museo | **Sm** | I musei |
| ***193*** | La musica | **Sf** | (Le) (musiche) |
| ***194*** | Il naso | **Sm** | I nasi |
| ***195*** | (Il) Natale | **Sm** | (I (Natale) |
| ***196*** | La nave | **Sf** | Le navi |
| ***197*** | La nazionalità | **Sf** | Le nazionalità |
| ***198*** | La nazione | **Sf** | Le nazioni |
| ***199*** | La ndrangheta | **Sf** | - - |
| ***200*** | Il negozio | **Sm** | I negozi |
| ***201*** | La neve | **Sf** | Le (nevi) |
| ***202*** | Il e La nipote | **Sm e Sf** | I e Le nipoti |
| ***203*** | La noia | **Sf** | - - |
| ***204*** | Il nome | **Sm** | I nomi |
| ***205*** | La/Il nonna/nonno | **Sf/Sm** | Le/I nonne/nonni |
| ***206*** | Il nord | **Sm** | - - |
| ***207*** | La notte | **Sf** | Le notti |
| ***208*** | (Il) novembre | **Sm** | - - |
| ***209*** | Il numero | **Sm** | I numeri |
| ***210*** | L' occhio | **Sm** | Gli occhi |
| ***211*** | L' olio | **Sm** | Gli (oli) |
| ***212*** | L' ora | **Sf** | Le ore |
| ***213*** | L' orecchio | **Sm** | Le orecchie (Sf) |
| ***214*** | L' oro | **Sm** | Gli (ori) |
| ***215*** | L' orologio | **Sm** | Gli orologi |
| ***216*** | L' ospedale | **Sm** | Gli ospedali |
| ***217*** | (L') ottobre | **Sm** | - - |
| ***218*** | L' ovest | **Sm** | - - |
| ***219*** | Il padre | **Sm** | I padri |
| ***220*** | Il paese | **Sm** | I paesi |

| | | | |
|---|---|---|---|
| ***221*** | Il palazzo | **Sm** | I palazzi |
| ***222*** | Il pallone | **Sm** | I palloni |
| ***223*** | Il pane | **Sm** | - - |
| ***224*** | Il pantalone | **Sm** | I pantaloni |
| ***225*** | Il papà | **Sm** | I papà |
| ***226*** | Il parco | **Sm** | I parchi |
| ***227*** | Il/La parente | **Sm/Sf** | I/Le parenti |
| ***228*** | La parola | **Sf** | Le parole |
| ***229*** | (La) Pasqua | **Sf** | Le (Pasque) |
| ***230*** | La passeggiata | **Sf** | Le passeggiate |
| ***231*** | La pasta | **Sf** | - - |
| ***232*** | La patata | **Sf** | Le patate |
| ***233*** | La paura | **Sf** | Le paure |
| ***234*** | Il pazzo | **Sm** | I pazzi |
| ***235*** | La penna | **Sf** | Le penne |
| ***236*** | Il pepe | **Sm** | - - |
| ***237*** | La persona | **Sf** | Le persone |
| ***238*** | Il pesce | **Sm** | I pesci |
| ***239*** | Il piatto | **Sm** | I piatti |
| ***240*** | La piazza | **Sf** | Le piazze |
| ***241*** | Il piede | **Sm** | I piedi |
| ***242*** | La pioggia | **Sf** | Le (piogge) |
| ***243*** | La pizza | **Sf** | Le pizze |
| ***244*** | La pizzeria | **Sf** | Le pizzerie |
| ***245*** | La politica | **Sf** | Le politiche |
| ***246*** | La polizia | **Sf** | Le polizie |
| ***247*** | Il poliziotto | **Sm** | I poliziotti |
| ***248*** | Il pollo | **Sm** | I polli |
| ***249*** | Il pomeriggio | **Sm** | I pomeriggi |
| ***250*** | Il pomodoro | **Sm** | I pomodori |
| ***251*** | Il ponte | **Sm** | I ponti |
| ***252*** | La porta | **Sf** | Le porte |
| ***253*** | Il porto | **Sm** | I porti |
| ***254*** | La possibilità | **Sf** | Le possibilità |
| ***255*** | La posta | **Sf** | Le poste |
| ***256*** | Il pranzo | **Sm** | I pranzi |
| ***257*** | Il presidente | **Sm** | I presidenti |
| ***258*** | Il prezzo | **Sm** | I prezzi |
| ***259*** | La primavera | **Sf** | Le primavere |
| ***260*** | Il problema | **Sm** | I problemi |
| | professore/professor | | professori/professore |
| ***261*** | Il/La essa | **Sm/Sf** | I/Le sse |
| ***262*** | La/Il ragazza/ragazzo | **Sf/Sm** | Le/I ragazze/ragazzi |
| ***263*** | Il re | **Sm** | I re |
| ***264*** | Il e La regista | **Sm e Sf** | I e Le registi e registe |
| ***265*** | La religione | **Sf** | Le religioni |

| | | | |
|---|---|---|---|
| ***266*** | La repubblica | **Sf** | Le repubbliche |
| ***267*** | Il ristorante | **Sm** | I ristoranti |
| ***268*** | Il ritorno | **Sm** | I ritorni |
| ***269*** | La rosa | **Sf** | Le rose |
| ***270*** | Il sabato | **Sm** | I sabati |
| ***271*** | Il salame | **Sm** | I salami |
| ***272*** | Il sale | **Sm** | I (sali) |
| ***273*** | Il salotto | **Sm** | I salotti |
| ***274*** | La scarpa | **Sf** | Le scarpe |
| ***275*** | La scuola | **Sf** | Le scuole |
| ***276*** | Il secolo | **Sm** | I secoli |
| ***277*** | Il secondo | **Sm** | I secondi |
| ***278*** | La sedia | **Sf** | Le sedie |
| ***279*** | La sera | **Sf** | Le sere |
| ***280*** | Il sesso | **Sm** | I (sessi) |
| ***281*** | La sete | **Sf** | - - |
| ***282*** | (Il) settembre | **Sm** | - - |
| ***283*** | La settimana | **Sf** | Le settimane |
| ***284*** | La sigaretta | **Sf** | Le sigarette |
| ***285*** | La/Il signora/signore | **Sf e Sm** | Le/I signore/signori |
| ***286*** | La sinistra | **Sf** | Le (sinistre) |
| ***287*** | Il sole | **Sm** | I (soli) |
| ***288*** | Il sonno | **Sm** | I (sonni) |
| ***289*** | La sorella | **Sf** | Le sorelle |
| ***290*** | Lo (spaghetto) | **Sm** | Gli spaghetti |
| ***291*** | Lo (spicciolo) | **Sm** | Gli spiccioli |
| ***292*** | Lo sport | **Sm** | Gli sport |
| ***293*** | Lo stadio | **Sm** | Gli stadi |
| ***294*** | La stagione | **Sf** | Le stagioni |
| ***295*** | La stanza | **Sf** | Le stanze |
| ***296*** | La stazione | **Sf** | Le stazioni |
| ***297*** | La storia | **Sf** | Le storie |
| ***298*** | La strada | **Sf** | Le strade |
| ***299*** | Lo straniero | **Sm** | Gli stranieri |
| ***300*** | Lo/La studente/studentessa | **Sm/Sf** | Gli/Le studenti/studentesse |
| ***301*** | Il sud | **Sm** | - - |
| ***302*** | La/Il suocera/suocero | **Sf/Sm** | Le/I suocere/suoceri |
| ***303*** | Il tavolo | **Sm** | I tavoli |
| ***304*** | Il taxi | **Sm** | I taxi |
| ***305*** | La tazza | **Sf** | Le tazze |
| ***306*** | Il teatro | **Sm** | I teatri |
| ***307*** | Il telefonino | **Sm** | I telefonini |
| ***308*** | Il telefono | **Sm** | I telefoni |
| ***309*** | La televisione | **Sf** | Le televisioni |
| ***310*** | Il tempo | **Sm** | I (tempi) |
| ***311*** | La testa | **Sf** | Le teste |

| | | | |
|---|---|---|---|
| ***312*** | Il tiramisù | **Sm** | I tiramisù |
| ***313*** | La torta | **Sf** | Le torte |
| ***314*** | Il (tortellino) | **Sm** | I tortellini |
| ***315*** | Il traffico | **Sm** | I (traffici) |
| ***316*** | Il tram | **Sm** | I tram |
| ***317*** | Il treno | **Sm** | I treni |
| ***318*** | Il e La turista | **Sm e Sf** | I e Le turisti e turiste |
| ***319*** | L' università | **Sf** | Le università |
| ***320*** | L' uomo | **Sm** | Gli uomini |
| ***321*** | La vacanza | **Sf** | Le vacanze |
| ***322*** | Il venerdì | **Sm** | I venerdì |
| ***323*** | Il vestito | **Sm** | I vestiti |
| ***324*** | Il vetro | **Sm** | I (vetri) |
| ***325*** | La via | **Sf** | Le vie |
| ***326*** | Il viaggio | **Sm** | I viaggi |
| ***327*** | La villa | **Sf** | Le ville |
| ***328*** | Il vino | **Sm** | I vini |
| ***329*** | Il visto | **Sm** | I visti |
| ***330*** | La vita | **Sf** | Le vite |
| ***331*** | La volta | **Sf** | Le volte |
| ***332*** | La/Lo zia/zio | **Sf/Sm** | Le/Gli zie/zii |
| ***333*** | Lo zucchero | **Sm** | Gli (zuccheri) |