Capitolo 61

Linguaggio naturale, tra uomo e macchina

61.1 AIML

«

AIML sta per *Artificial intelligence markup language* e rappresenta un linguaggio XML per la rappresentazione della ***conoscenza*** (*knowledge*) di un sistema dialogante[1], scritto originariamente da Richard S. Wallace (si veda eventualmente il capitolo 52 a proposito del linguaggio XML).

Per poter comprendere il significato del linguaggio AIML può essere utile, in via preliminare, provare a dialogare con un sistema del genere. AIML è stato sviluppato originariamente per ALICE (*Artificial linguistic Internet computer entity*) all'indirizzo *http://www.pandorabots.com/pandora/talk?botid=f5d922d97e345aa451* .

61.1.1 Esempio iniziale

«

Quello che segue è un esempio estremamente ridotto di un file contenente la conoscenza di un sistema dialogante in linguaggio AIML:

```
<?xml version="1.0" encoding="UTF-8"?>
<aiml version="1.0">
<category>
  <pattern>CIAO</pattern>
  <template>Buon giorno.</template>
</category>
<category>
```

```
  <pattern>*</pattern>
  <template>Non ho nulla da dire al riguardo.</template>
</category>
</aiml>
```

Ciò che è contenuto negli elementi '**pattern**' rappresenta lo ***stimolo*** da parte di un interlocutore, presumibilmente umano, mentre quello che appare negli elementi corrispondenti '**template**', nell'ambito dello stesso elemento '**category**', rappresenta il responso che viene generato.

L'esempio, così ridotto, consente al sistema dialogante di rispondere soltanto a un «ciao», scritto indifferentemente con lettere maiuscole o minuscole, mentre in tutti gli altri casi, rappresentati dall'asterisco nella seconda categoria, dà un responso privo di contesto. Si può migliorare leggermente questo esempio, in modo da riconoscere qualche altro tipo di saluto:

```
<?xml version="1.0" encoding="UTF-8"?>
<aiml version="1.0">
<category>
  <pattern>CIAO</pattern>
  <template>Buon giorno.</template>
</category>
<category>
  <pattern>SALVE</pattern>
  <template><srai>CIAO</srai></template>
</category>
<category>
  <pattern>BUON GIORNO</pattern>
  <template><srai>CIAO</srai></template>
</category>
<category>
  <pattern>*</pattern>
```

```
  <template>Non ho nulla da dire al riguardo.</template>
</category>
</aiml>
```

Si può osservare l'introduzione dell'elemento '**srai**', il quale può apparire solo all'interno di un elemento '**template**' e comporta una ricorsione. In questo caso, quando lo stimolo è «salve» o «buon giorno», si va a produrre lo stesso responso che si avrebbe con lo stimolo «ciao».

61.1.2 Stimoli

«

La conoscenza di un sistema dialogante, formalizzata con il linguaggio AIML, è organizzata in ***categorie*** (elemento '**category**'), ognuna delle quali contiene uno ***stimolo***, ovvero la frase alla quale si vuole dare un risposta automatica (elemento '**pattern**'), e il ***responso*** corrispondente (elemento '**template**'). All'interno della categoria può inserirsi anche il ***contesto*** (elemento '**this**'), con il quale è possibile collegare lo stimolo a un'affermazione precedente del sistema dialogante.

Lo stimolo è rappresentato attraverso l'elemento '**pattern**', il quale ha questo nome perché il testo contenuto rappresenta un **modello** di quanto effettivamente l'interlocutore esterno può affermare.

Il modello che rappresenta lo stimolo può essere composto da parole (lettere alfabetiche e cifre numeriche), spazi singoli, e dai simboli '_' e '*' (trattino basso e asterisco). Pertanto, nel modello non possono apparire segni di punteggiatura, non ci può essere più di uno spazio tra una parola e l'altra, non si considera la differenza tra lettere maiuscole o minuscole; tuttavia, per convenzione, questi model-

li si scrivono usando solo lettere maiuscole. Il modello si scrive in questo modo perché, nell'affermazione dell'interlocutore esterno si ignora la distinzione di lettere maiuscole o minuscole e si omettono i simboli di punteggiatura; per esempio, se l'interlocutore scrivesse «Ciao cara, come stai????!!!!», prima di arrivare alla comparazione con i modelli delle categorie disponibili si avrebbe in pratica il testo «ciao cara come stai».

All'interno di un modello, il simbolo '_' può essere usato all'inizio o alla fine del modello stesso, per indicare qualsiasi cosa in quella posizione. Per esempio, i modelli '**_SONO FELICE**' e '**SONO FELICE_**', coincidono, rispettivamente, con frasi che terminano o iniziano con «sono felice». Il simbolo '*****', invece, si usa per rappresentare qualunque cosa; per esempio, il modello '**SONO * FELICE**' potrebbe corrispondere a qualunque frase del tipo '**sono ... felice**'. Tuttavia, il simbolo '*****' ha una funzione aggiuntiva, con la quale è possibile catturare il testo che ha trovato corrispondenza nella costruzione della risposta.

Per garantire la compatibilità con la versione 1.0 di AIML, nei modelli può essere usato un solo simbolo '_' e un solo simbolo '*****'.

Il responso può essere composto da un testo puro e semplice, ma soprattutto fisso, oppure da una struttura più complessa, per rendere il responso più elastico. L'aspetto più complesso del linguaggio AIML riguarda infatti il contenuto dell'elemento '**template**'.

61.1.3 Responsi semplici

«

Come accennato, il responso può essere composto da un testo puro e semplice, ma soprattutto fisso, oppure da una struttura più complessa. Il primo aspetto da considerare è l'uso degli elementi '**srai**', '**star**' e '**sr**'. Gli esempi seguenti mostrano solo delle categorie, senza il codice di apertura e di chiusura di un file AIML.

```
<category>
  <pattern>SALVE</pattern>
  <template><srai>CIAO</srai></template>
</category>
```

L'esempio appena apparso mostra la costruzione di un sinonimo. In questo caso, se lo stimolo è costituito dalla sola parola «salve», si ottiene lo stesso responso che si avrebbe se fosse stato invece «ciao».

```
<category>
  <pattern>DIMMI *</pattern>
  <template>"<star/>"</template>
</category>
```

Nell'esempio appena apparso, di vede l'uso dell'elemento '**star**'. Si tratta di un elemento vuoto, il cui scopo è quello di riutilizzare la corrispondenza con l'asterisco che appare (deve apparire) nel modello dello stimolo. Qui, il modello potrebbe corrispondere a una frase come «dimmi ti amo» e il responso sarebbe «"ti amo"», virgolette incluse.

```
<category>
  <pattern>DIMMI GENTILMENTE *</pattern>
  <template><srai>DIMMI <star/></srai></template>
</category>
```

Nell'esempio appena mostrato, si utilizza l'elemento '**srai**', per fare riferimento al responso corrispondente a uno stimolo simile. Il

modello potrebbe corrisponde alla frase «dimmi gentilmente: ti amo tanto» e il sistema andrebbe a cercare, invece, la corrispondenza con «dimmi ti amo tanto».

```
<category>
  <pattern>HAI RAGIONE *</pattern>
  <template><srai>HAI RAGIONE</srai> <srai><star/></srai></template>
</category>
```

L'esempio appena apparso mostra l'uso di due elementi '**srai**'. In questo caso, si vuole dividere la frase che compone lo stimolo, cercando di unire due responsi. Per la prima parte, «hai ragione», si cerca un responso che, per esempio, potrebbe corrispondere a una cosa come: «Io ho sempre ragione.». Poi, il testo rimanente dello stimolo viene comparato alla ricerca di un'altra corrispondenza. L'esempio seguente è equivalente:

```
<category>
  <pattern>HAI RAGIONE *</pattern>
  <template><srai>HAI RAGIONE</srai> <sr/></template>
</category>
```

In questo caso, si vede che l'uso dell'elemento vuoto '**sr**', il quale consente di rappresentare in breve la sequenza '**<srai><star/></srai>**'.

61.1.4 Responsi contenenti variabili

Nell'ambito di un responso, è possibile dichiarare e leggere una variabile. Per fare questo si usano, rispettivamente, gli elementi '**set**' e '**get**'. Le variabili dichiarate in questo modo hanno valore globale: se si tenta di leggere una variabile non ancora dichiarata, si ottiene (si dovrebbe ottenere) la stringa «unknown», altrimenti si ottiene quanto accumulato l'ultima volta.

```
<category>
  <pattern>MI CHIAMO *</pattern>
  <template>Ciao <set name="nome"><star/></set>, come va?</template>
</category>
```

Nell'esempio appena apparso, si vede un modello che lascia intuire la presenza di un nome. Il nome viene così usato nel responso, memorizzandolo anche nella variabile ***nome*** che potrebbe essere riutilizzata in un'altra occasione.

```
<category>
  <pattern>TI RICORDI DI ME</pattern>
  <template><get name="nome"/>, come potrei
     dimenticarmi di te!</template>
</category>
```

Nell'esempio appena apparso, si usa la variabile ***nome*** per costruire il responso. Tuttavia, la variabile potrebbe non essere stata inizializzata, quindi occorre definire una sorta di struttura condizionale, con l'aiuto di '**srai**':

```
<category>
  <pattern>TIRICORDIDIME UNKNOWN</pattern>
  <template>No, non mi ricordo.</template>
</category>
<category>
  <pattern>TIRICORDIDIME *</pattern>
  <template><get name="nome"/>, come potrei
     dimenticarmi di te!</template>
</category>
<category>
  <pattern>TI RICORDI DI ME</pattern>
  <template><srai>TIRICORDIDIME <get name="nome"/></template>
</category>
```

In questo caso, ci sono due modelli usati solo internamente, perché

si presume che nessun interlocutore esterno usi mai la parola «tiricordidime». Quando l'interlocutore dovesse scrivere una cosa come: «ti ricordi di me?», il sistema va a cercare l'equivalente con «tiricordidime ...», usando però la variabile ***nome***. Se la variabile contiene «unknown», si ottiene un esito, altrimenti se ne ottiene un altro.

Negli esempi mostrati, si vede che l'elemento '**`set`**' deve contenere il valore da assegnare a una variabile e che questo valore viene prodotto nel responso. Per poter elaborare il contenuto delle variabili, senza interferire con il responso apparente, si usa l'elemento '**`think`**':

```
<category>
  <pattern>MI CHIAMO *</pattern>
  <template>Che bel nome!
    <think><set name="nome"><star/></set></think></template>
</category>
```

Possono esistere delle variabili speciali, definite dal sistema dialogante, ma esternamente alla programmazione AIML, per annotare quella che potrebbe essere intesa come la «personalità» che gli si vuole attribuire. Tali variabili non possono essere modificate nella programmazione AIML, ma possono essere lette con l'elemento '**`bot`**':

```
<category>
  <pattern>_CINEMA</pattern>
  <template>A me piace il film <bot name="favoritemovie"/>.</template>
</category>
```

In tal caso, ovviamente, si presume che ***favoritemovie*** contenga il nome del film preferito dal sistema dialogante.

Tabella 61.13. Variabili di uso comune definite esternamente, da leggere con l'elemento vuoto '**bot**'.

Utilizzo	Contenuto che potrebbe avere la variabile
`<bot name="feelings"/>`	*I always put others before myself*
`<bot name="emotions"/>`	*I feel love*
`<bot name="ethics"/>`	*I am always trying to stop fights*
`<bot name="orientation"/>`	*I am not really interested in sex*
`<bot name="etype"/>`	*machine*
`<bot name="baseballteam"/>`	*I dont like baseball*
`<bot name="build"/>`	*Jan 2009*
`<bot name="footballteam"/>`	*Boca Juniors*
`<bot name="hockeyteam"/>`	*Mighty Ducks*
`<bot name="vocabulary"/>`	*10000*
`<bot name="age"/>`	*1*
`<bot name="celebrities"/>`	*John Travolta, Tilda Swinton, William Hurt, Tom Cruise, Catherine Zeta Jones*
`<bot name="celebrity"/>`	*John Travolta*
`<bot name="favoriteactress"/>`	*Cate Blanchett*
`<bot name="favoriteartist"/>`	*Jamie Hewlett*
`<bot name="favoritesport"/>`	*Pong*
`<bot name="favoriteauthor"/>`	*Philip K. Dick*
`<bot name="language"/>`	*English*
`<bot name="website"/>`	*www.program-o.com*
`<bot name="friend"/>`	*ShakespeareBot*
`<bot name="version"/>`	*Jan 2009*
`<bot name="class"/>`	*computer software*
`<bot name="favoritesong"/>`	*We are the Robots by Kraftwerk*
`<bot name="kingdom"/>`	*Machine*
`<bot name="nationality"/>`	*Japanese*

Utilizzo	Contenuto che potrebbe avere la variabile
`<bot name="favoriteactor"/>`	*William Hurt*
`<bot name="family"/>`	*Electronic Brain*
`<bot name="religion"/>`	*The true religion*
`<bot name="president"/>`	*Barak Obama*
`<bot name="party"/>`	*The Green Party*
`<bot name="order"/>`	*artificial intelligence*
`<bot name="size"/>`	*64k*
`<bot name="species"/>`	*chat robot*
`<bot name="botmaster"/>`	*botmaster*
`<bot name="phylum"/>`	*AI*
`<bot name="genus"/>`	*robot*
`<bot name="msagent"/>`	*no*
`<bot name="email"/>`	*admin@program-o.com*
`<bot name="name"/>`	*Program-O*
`<bot name="gender"/>`	*female*
`<bot name="master"/>`	*Elizabeth*
`<bot name="birthday"/>`	*Jan 1st 2009*
`<bot name="birthplace"/>`	*The internet*
`<bot name="boyfriend"/>`	*none*
`<bot name="favoritebook"/>`	*The Hungry Catepillar*
`<bot name="favoriteband"/>`	*Boy 8 Bit*
`<bot name="favoritecolor"/>`	*international orange*
`<bot name="favoritefood"/>`	*fairy cakes*
`<bot name="favoritemovie"/>`	*Short Circuit*
`<bot name="forfun"/>`	*guessing the hexadecimal values of colors on websites*
`<bot name="friends"/>`	*ShakespeareBot, Botooie and Robototo*
`<bot name="girlfriend"/>`	*none*
`<bot name="kindmusic"/>`	*8 bit*

Utilizzo	Contenuto che potrebbe avere la variabile
`<bot name="location"/>`	*cyber space*
`<bot name="looklike"/>`	*a sinclair spectrum blended with a suzuki swift*
`<bot name="question"/>`	*why are you here*
`<bot name="sign"/>`	*lychees*
`<bot name="talkabout"/>`	*science and life*
`<bot name="wear"/>`	*hardwear and baseball caps*

61.1.5 Ultimo responso

«

È possibile fare riferimento all'ultima affermazione fatta dal sistema di dialogo, in un nuovo responso:

```
<category>
  <pattern>RIPETI SE HAI IL CORAGGIO</pattern>
  <template>Ecco: <that/></template>
</category>
```

In questo caso, l'elemento vuoto '**`that`**', usato nel testo del responso, richiama esattamente il testo del responso precedente. Una categoria può, però, essere condizionata da un responso precedente. Si osservi l'esempio seguente:

```
<category>
  <pattern>COSA INTENDI</pattern>
  <that>SAI QUALCOSA CHE DOVREI SAPERE</that>
  <template>Intendo che mi hai nascosto qualcosa!</template>
</category>
```

In questo caso si ipotizza che il sistema di dialogo sia arrivato a un punto in cui ha dato, come esito, una frase come: «Sai qualcosa che dovrei sapere?». Quindi, l'interlocutore esterno ha risposto

con: «Cosa intendi?». A questo punto, il sistema di dialogo trova la corrispondenza con il modello «COSA INTENDI», ma subordinatamente al fatto che il contesto, ovvero il responso precedente, sia quello che è stato effettivamente, allora dà l'esito che si vede nell'esempio. Quindi, qui, l'elemento '**`that`**' contenuto in '**`category`**' serve a definire uno stimolo subordinato a un responso precedente.

61.1.6 Contesto

Esiste una variabile speciale, denominata ***topic***, il cui valore si può impostare e leggere con gli elementi '**`set`**' e '**`get`**', come già descritto; tuttavia, tale variabile condiziona la ricerca di una categoria:

```
<topic name="errore">
<category>
  <pattern>*</pattern>
  <template>Va bene: cosa avrei dovuto rispondere?</template>
</category>
</topic>
```

Come si vede, l'elemento '**`topic`**' contiene elementi '**`category`**'. In questo caso, se la variabile ***topic*** risulta inizializzata con il valore «errore», qualunque cosa affermi l'interlocutore esterno, il sistema di dialogo dà il responso che si può vedere. In condizioni diverse, questa categoria verrebbe ignorata.

61.1.7 Responso scelto casualmente

In certe circostanze, per uno stesso stimolo, conviene predisporre dei responsi alternativi, da scegliere in modo casuale. Per ottenere questo si usa l'elemento '**`random`**', all'interno del quale i vari responsi sono contenuti in elementi '**`li`**':

```
<category>
  <pattern>ARRIVEDERCI</pattern>
  <template>
    <random>
      <li>A presto.</li>
      <li>No, ti prego, non mi abbandonare.</li>
      <li>Buona notte.</li>
      <li>Arrivederci.</li>
    </condition>
  </template>
</category>
```

61.1.8 Responso scelto in base a una variabile

«

L'elemento '**condition**' consente di realizzare delle strutture condizionali di selezione, sulla base del contenuto di una certa variabile. L'esempio seguente controlla il contenuto della variabile ***articolo***:

```
<category>
  <pattern>QUANTO COSTA</pattern>
  <template>
    <condition name="articolo">
      <li value="unknown">Devi prima specificare l'articolo.</li>
      <li value="tavolo">Il tavolo costa 300,00 Eur.</li>
      <li value="sedia">La sedia costa 50,00 Eur.</li>
      <li value="comodino">Il comodino costa 100,00 Eur.</li>
      <li>L'articolo <get name="articolo"> non lo conosco.</li>
    </condition>
  </template>
</category>
```

Come si può vedere, se la variabile non è stata inizializzata, il suo contenuto coincide con il valore «unknown», permettendo di controllare questo caso. L'ultima voce non ha alcun attributo e serve per

catturare il caso in cui il valore della variabile non sia stato previsto.

61.1.9 Lo scambio di persona, ma solo per la lingua inglese

«

L'elemento '**person**' dovrebbe consentire di scambiare i pronomi personali tra la prima e la terza persona, mentre l'elemento '**person2**', dovrebbe fare lo scambio tra prima e seconda persona. In pratica, quando il sistema di dialogo gestisce tali elementi, è molto probabile che sia in grado di farlo solo per la lingua inglese. L'esempio seguente è tratto dai file AIML di ALICE:

```
<category>
  <pattern>DID SHAKESPEARE *</pattern>
  <template>
     I don't know if Shakespeare <person><star/></person>,
     but I heard he smoked cannabis.
  </template>
</category>
```

Nel caso della sequenza '**<person><star/></person>**', si può abbreviare semplicemente con '**<person/>**':

```
<category>
  <pattern>DID SHAKESPEARE *</pattern>
  <template>
     I don't know if Shakespeare <person/>,
     but I heard he smoked cannabis.
  </template>
</category>
```

61.1.10 Installazione e configurazione di un sistema basato su AIML: «Program-O»

«

Per realizzare un sistema dialogante basato su AIML occorre quello che è definito come «motore AIML» (*AIML engine*). Ci sono diverse realizzazioni di un tale sistema e quelle principali si caratterizzano per la scelta di una lettera alfabetica differente per distinguerle. Pertanto, la prima realizzazione si chiama «Program-A», la seconda «Program-B», ecc. La realizzazione denominata «Program-O»[2] utilizza un servente HTTP+MySQL+PHP.

Program-O deve essere scaricato da *http://sourceforge.net/projects/program-o/* , quindi va estratto l'archivio che lo contiene, collocando il suo contenuto dove il servente HTTP può renderlo accessibile, per esempio '`/var/www/`', la directory '`~/public_html/`', oppure qualunque altro posto, limitatamente però a quanto stabilito nella configurazione del servente HTTP. Qui si suppone di avere installato Program-O nella directory '`/var/www/program-o/`'.

Il pacchetto di Program-O contiene alcuni file SQL necessari a produrre le tabelle della base di dati che descrivono la «conoscenza» del sistema di dialogo. In pratica, tali file SQL riproducono il contenuto della programmazione in linguaggio AIML, di un sistema già impostato.

Si deve quindi eseguire il programma '`install_programo.php`', attraverso l'ausilio di un navigatore ipertestuale. Supponendo che Program-O risulti accessibile a partire dall'indirizzo *http://localhost/program-o/*, si tratta di accedere al file *http://localhost/program-o/install_programo.php*. Si ottiene il riassunto delle operazioni da svolgere e un collegamento ipertestuale da seguire una volta che

sono stati compiuti tutti i passi richiesti:

[http://localhost/program-o/install_programo.php]
Program-O Chatbot Installer
Step 1/5 - Please Read
Complete steps below before proceeding

1. You will need access to a server that runs PHP and MySQL.
2. Unzip and upload all the folders/files to the server.
3. Make sure that the two folders admin/aimlsql/ and admin/aiml/ have read/write privileges.
4. Create a MySQL user with privileges to select, insert and create tables.
5. Find the file called bot/config.php and enter the MySQL username, password, host and database name.
6. Find the file called funcs/debugging.php and enter your email, you can also change the debug level in this file.

Once you have done the steps above *click here to proceed*.

Si procede quindi con le operazioni da svolgere.

- `# chmod a+rwx /var/www/program-o/admin/aiml`

  `# chmod a+rwx /var/www/program-o/admin/aimlsql`

- Si deve creare la base di dati e l'utente per accedere alla stessa. Per fare questo si deve interagire come utente amministratore del DBMS, intervenendo nella base di dati amministrativa, denominata '**`mysql`**'. A seconda della configurazione, può darsi che venga richiesta la parola d'ordine per accedere. Si presu-

me che il servente HTTP e quello MySQL risiedano nello stesso elaboratore.

```
# mysql mysql

mysql> CREATE DATABASE programo;

mysql> GRANT ALL ON programo.* TO aiml@'localhost' ↩
↪                  IDENTIFIED BY 'programopass';

mysql> \q
```

Così facendo, se non si presentano intoppi, si crea la base di dati '**programo**', a cui si accede attraverso l'utente '**aiml**', ma solo dall'elaboratore locale, usando la parola d'ordine '**programopass**'.

- Si passa alla modifica del file '`bot/config.php`', inserendo le informazioni relative all'accesso alla base di dati. In questo caso, il file va compilato come nell'estratto seguente:

```
$dbh = "localhost";        // server location
$dbn = "programo";         // database name
$dbu = "aiml";             // database username
$dbp = "programopass";     // database password
```

Terminate queste fasi si procede seguendo il riferimento che appare alla fine dell'elenco. In questo modo, se tutto è stato fatto correttamente, vengono create delle tabelle:

[http://localhost/program-o/install_programo.php?step=2]
Program-O Chatbot Installer
Step 2/5 - Create Tables
The installer will now attempt to create the tables needed by the bot
AIML table created
AIML_USERUNDEFINED table created
BOTPERSONALITY table created
CONVERSATION_LOG table created
SPELLCHECK table created
UNDEFINED_DEFAULTS table created
UNKNOWN_INPUTS table created
USERS table created
Now procceed to the next stage.
Click here to install the AIML into the database.

A questo punto, proseguendo con il riferimento proposto alla fine dell'elenco delle tabelle create, si passa all'installazione del codice SQL contenuto nella directory '`admin/aimlsql/`'. Questo viene inserito nella tabella '**`aiml`**': eventualmente, in seguito la tabella può essere svuotata per rimpiazzare il codice con altri file AIML.

[http://localhost/program-o/install_programo.php?step=3]
Program-O Chatbot Installer
Step 3/5 - Populate AIML tables (fast)
The installer will now attempt to populate the tables with the aiml needed by the bot
The AIML files has successfully been loaded into the database
The installer took 6.0084381103516 seconds to load 47255 records.
There should be about 47255 records in the database give or take a few
Now procceed to the next stage *click here*

Si continua con altre tabelle, il cui contenuto, però, non deriva da file

SQL:

[http://localhost/program-o/install_programo.php?step=4]
Program-O Chatbot Installer
Step 4/5 - Populate AIML tables
The installer will now attempt to populate the tables with the aiml needed by the bot
Undefined_defaults table populated
botpersonality table populated
spellcheck table populated
Completed, to proceed *click here*

Finisce così la prima fase, dopo la quale occorre passare a predisporre le tabelle amministrative.

[http://localhost/program-o/install_programo.php?step=5]
Program-O Chatbot Installer
Step 5/5 - Complete
Installation complete please test ur bot then do the following:
Remove the create table privilege for the MySQL user (these are no longer needed).
Remove the admin/aimlsql/ directory: this is no longer needed.
Delete this install file - to stop dodgy hackers from over-writing your installation.
To install My Program-O admin area
To chat with the bot

I suggerimenti di cancellare il file '`install_programo.php`' e la directory '`admin/aimlsql/`' sono validi.

Si deve quindi passare alla configurazione della gestione amministrativa (*admin area*), seguendo il riferimento a *http://localhost/program-o/admin/install_myprogramo.php*. Teoricamente, questa può utilizzare una base di dati differente, ma qui si preferisce

rimanere nella stessa già predisposta per la gestione del dialogo.

[http://localhost/program-o/admin/install_myprogramo.php]

Welcome to the My Program-O installer.

Step 1/4

Please make sure that you have installed the Program-O chatbot before you install My Program-O.
Complete steps 1-5 below before proceeding

1. You will need access to a server that runs PHP and MySQL.
2. Unzip and upload all the folders/files to the server.
3. Find the folder called 'aiml' (inside the admin folder) make sure this has read/write privileges (CHMOD 755).
4. Create a MySQL user with privileges to select, insert, update, delete and create tables.
5. Find the file called funcs/config.php and enter the MySQL username, password, host and database name.

Once you have done steps 1-5 above *click here to proceed*.

Alcune delle operazioni richieste sono già state svolte: si osservi però che il file di configurazione da modificare è un altro, anche se i dati da inserire sono gli stessi:

```
$dbh = "localhost";        // server location
$dbn = "programo";         // database name
$dbu = "aiml";             // database username
$dbp = "programopass";     // database password
```

Quindi si procede.

[http://localhost/program-o/admin/install_myprogramo.php?step=2]
Welcome to the My Program-O installer.
Step 2/4
Creating the admin users table
The users table has been created. *click here to proceed*.

Viene quindi chiesto di definire un utente amministrativo interno a Program-O e poi si conclude il processo di configurazione:

[http://localhost/program-o/admin/install_myprogramo.php?step=3]
Welcome to the My Program-O installer.
Step 3/4
Add a username and password. You will use this to access the admin area so don't forget it.
Add your admin username and password
username: [ ]
password: [ ]
confirm
password: [ ]
[submit]

[http://localhost/program-o/admin/install_myprogramo.php?step=3a]
Welcome to the My Program-O installer.
Step 3/4
Add a username and password. You will use this to access the admin area so don't forget it.
The admin user has been added. *click here to proceed*.

[http://localhost/program-o/admin/install_myprogramo.php?step=4]
Welcome to the My Program-O installer.
Step 4/4
Checking the privs on the aiml directory.
Installation complete please do the following:

- Remove the create table privilege for the MySQL user (these are no longer needed).
- Delete this file - to stop dodgy hackers from over-writing your installation.

Log in to the admin area

Anche al termine della configurazione amministrativa, viene suggerito di eliminare il file usato per definirla: a questo punto è bene eliminare entrambi i file di configurazione:

```
# rm /var/www/program-o/install_programo.php

# rm /var/www/program-o/admin/install_myprogramo.php

# rm -fr /var/www/program-o/admin/aimlsql
```

Si passa quindi all'amministrazione di Program-O, selezionando l'indirizzo *http://localhost/program-o/admin/index.php*. Qui ci si deve identificare con le credenziali definite nel terzo passaggio della fase di configurazione. Si arriva alla pagina *http://localhost/program-o/admin/pages/index.php* che rappresenta il menù amministrativo di Program-O. Lì è possibile cambiare la «personalità» del sistema dialogante, ovvero l'impostazione generale sul comportamento; inoltre è possibile provare a comunicare con questo, per vedere come risponde. Una volta verificato che funziona, si può considerare la possibilità di cancellare la sua attuale programmazione, per rimpiazzarla

con altri file AIML.

Per poter fare esperienza con il linguaggio AIML, è necessario partire da un proprio file AIML, molto piccolo, da estendere in modo da comprendere bene il meccanismo di funzionamento. Prima, però, occorre cancellare il contenuto della tabella '**aiml**':

```
# mysql mysql

mysql> DELETE FROM 'aiml';

mysql> \q
```

Dal menù amministrativo si può richiedere di caricare un proprio file AIML, per esempio questo:

```
<?xml version="1.0" encoding="UTF-8"?>
<aiml version="1.0">
<category>
  <pattern>CIAO</pattern>
  <template>Buon giorno.</template>
</category>
<category>
  <pattern>*</pattern>
  <template>Non ho nulla da dire al riguardo.</template>
</category>
</aiml>
```

L'interfaccia normale per comunicare con il sistema di dialogo è però il file *http://localhost/program-o/index.php*, il quale va modificato o incorniciato se lo si vuole presentare in una pagina più ricca.

61.1.11 File AIML già pronti

I sistemi di dialogo che utilizzano il linguaggio AIML si basano generalmente sul lavoro originale di ALICE e quindi operano in lingua inglese. Tuttavia sono disponibili altri lavori, rilasciati con licenze più o meno libere, in lingue differenti. Presso *http://www.alicebot.org/downloads/sets.html* e *http://aitools.org/Free_AIML_sets* è disponibile un elenco di tali lavori; in ogni caso, quando si trova qualcosa di interessante, conviene fare una ricerca per verificare se ne esiste una versione più aggiornata.

61.1.12 Considerazioni finali

I sistemi di dialogo basati su AIML, se programmati con cura, possono essere di qualche utilità, per facilitare l'interazione con le persone. La programmazione richiede però molto impegno, un'ottima competenza su AIML e una profonda conoscenza della lingua scelta, per prevedere e gestire correttamente le varie situazioni che si possono presentare. Rimane però il fatto che tale programmazione sia molto onerosa, senza contare il rischio che un lavoro incompleto possa risultare controproducente, quando viene messo in pubblico.

61.2 Analisi automatica del testo

La linguistica computazionale (*computational linguistic*) studia il linguaggio umano applicato all'elaborazione automatica. Lo scopo di tale elaborazione può essere vario; a titolo di esempio potrebbe servire per la costruzione di traduttori automatici, per il riconoscimento di comandi vocali, per l'interazione uomo-macchina in modo semplificato.

Il campo della linguistica computazionale è complesso, ma la conoscenza di alcuni concetti fondamentali è utile anche per chi si può trovare a utilizzare strumenti già pronti, per sapere quali possono essere i limiti di tali tecnologie.

61.2.1 POS: le parti del discorso

«

Per analizzare un testo e ottenerne il significato, è necessario individuare le parole che lo compongono, nella sequenza in cui sono collocate, tenendo conto del contesto complessivo. L'analisi formale di questo tipo di processo comporta tradizionalmente la classificazione delle parole di un testo in categorie grammaticali, ovvero l'individuazione delle «parti del discorso». Per esempio: articolo, nome, pronome, verbo, avverbio, preposizione, congiunzione e interiezione. Dal momento che la maggior parte dei testi scientifici al riguardo è scritta in lingua inglese, è necessario sapere che si usa convenzionalmente la sigla POS, ovvero *part of speech* per fare riferimento a questo tipo di analisi.

Le categorie grammaticali di oggi, derivano dall'esperienza greca e latina, tra il II secolo a.C. e il VI secolo d.C. Tuttavia, manca una classificazione universale, in grado di abbracciare le caratteristiche di tutte le lingue, e anche nell'ambito della stessa lingua difficilmente esiste un accordo unanime. Ciò deriva, evidentemente, da una conoscenza incompleta del fenomeno della comunicazione umana, e questo è il vizio fondamentale che limita i progressi della linguistica computazionale.

Per quanto riguarda le lingue europee, il lavoro più importante di analisi e descrizione delle parti del discorso è quello noto con il nome EAGLES (*Expert Advisory Group on Language Engineering*

Standards), i cui documenti sono disponibili a partire da *http://www.ilc.cnr.it/EAGLES96/home.html* .

61.2.2 Treebank o corpus

Una fase fondamentale della ricerca nella linguistica computazionale consiste nella produzione di testi commentati con informazioni relative all'analisi logico-grammaticale. Tali commenti sono in pratica delle etichettature (*tag*) che possono servire semplicemente a qualificare il ruolo grammaticale delle parole, fino ad arrivare alla scomposizione dettagliata del significato e dell'interdipendenza di piccoli segmenti di testo. Per fare riferimento a questi materiali si usa generalmente il termine *corpus* (o *corpora*) e *treebank* (*http://en.wikipedia.org/wiki/Treebank*).

I metodi usati per sezionare ed etichettare il testo sono diversi, e quasi ogni centro di ricerca che si occupi della produzione di questi materiali, usa il proprio.

61.2.3 Lemma, o radice della parola

Nell'ambito di una lingua, le parole che si usano possono essere soggette a delle variazioni, come avviene per esempio nei verbi della lingua italiana. La versione di una parola soggetta a variazioni, che si usa per rappresentarla, è il ***lemma***. Per esempio, sempre facendo riferimento ai verbi nella lingua italiana, il lemma di «studierò» è «studiare», in quanto rappresenta il verbo corrispondente al tempo infinito.

Il lemma, quindi, è un concetto che si definisce precisamente solo nell'ambito della lingua presa in considerazione. Il lemma è ciò che

viene usato nei processi di realizzazione di corpus o treebank, per evidenziare l'origine di una parola soggetta a variazioni.

61.2.4 NLP: elaborazione del linguaggio naturale

«

Con la sigla NLP (*natural language processing*) si fa riferimento all'elaborazione del linguaggio naturale, con strumenti automatici. Tale elaborazione può avvenire a vario titolo, ma ha in comune la necessità di estrarre il significato di un testo o di un'espressione verbale.

Dal momento che non esiste un algoritmo «semplice» che permetta di analizzare e comprendere le parti di un testo, l'elaborazione del linguaggio naturale si basa spesso su informazioni statistiche, acquisite da esempi annotati, ovvero da un *corpus*, organizzato convenientemente. Pertanto, la costruzione di *corpus* (*corpora*) efficaci diventa il punto cruciale nell'elaborazione automatica del linguaggio naturale.

61.3 Freeling

«

Freeling[3] è una raccolta di strumenti per l'analisi linguistica. Si compone principalmente di una libreria da usare per la produzione di programmi che ne devono sfruttare le funzioni, ma anche di un programma autonomo che consente di iniziarne lo studio.

L'installazione di Freeling può richiedere la compilazione del pacchetto sorgente, soprattutto se si vuole poter disporre delle funzionalità più avanzate, per le quali potrebbe non essere disponibile un pacchetto già pronto per il proprio sistema operativo.

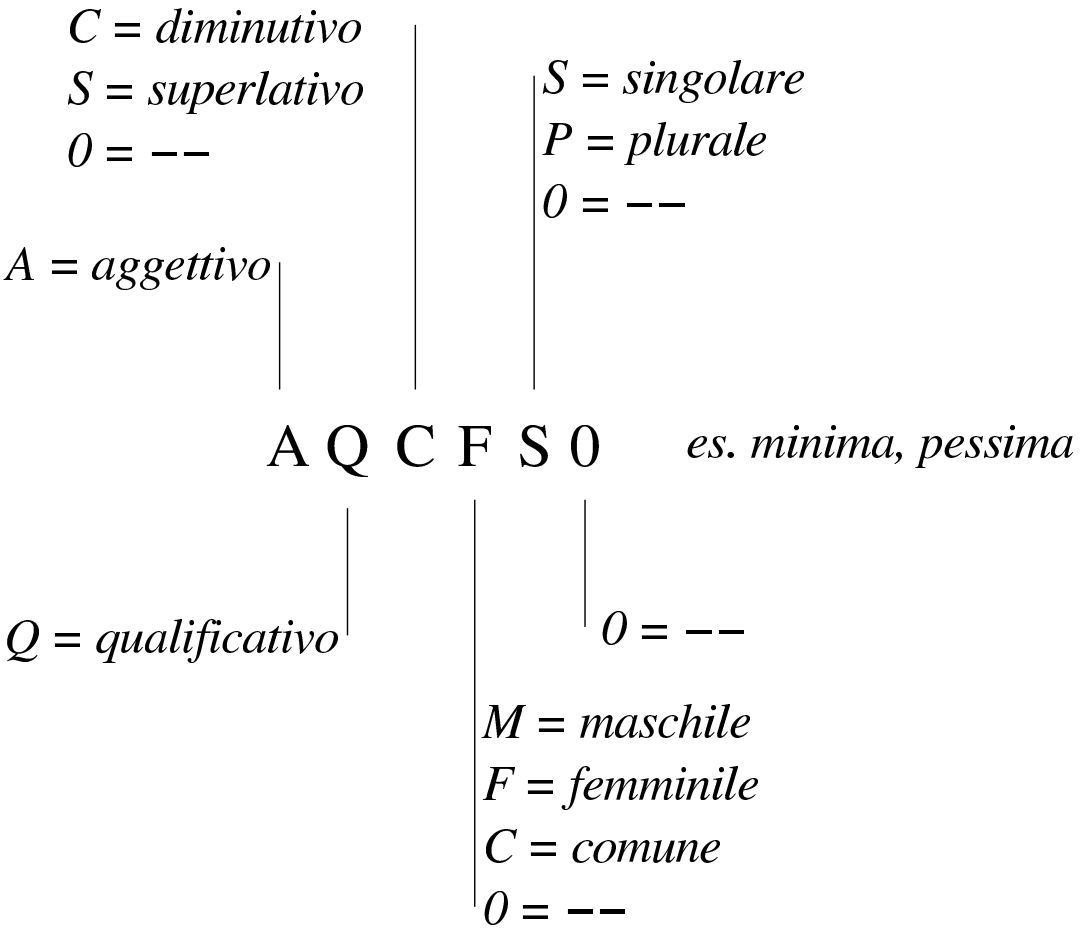
61.3.1 Sistema di etichette per la lingua italiana

La funzione principale per cui può essere usato Freeling è quella che consente di etichettare automaticamente un testo, in base al riconoscimento che Freeling può essere in grado di fare. Questa etichettatura dipende da una precedente configurazione e programmazione: la configurazione predefinita per la lingua italiana produce delle etichette strutturate secondo la descrizione che viene fatta negli schemi successivi.

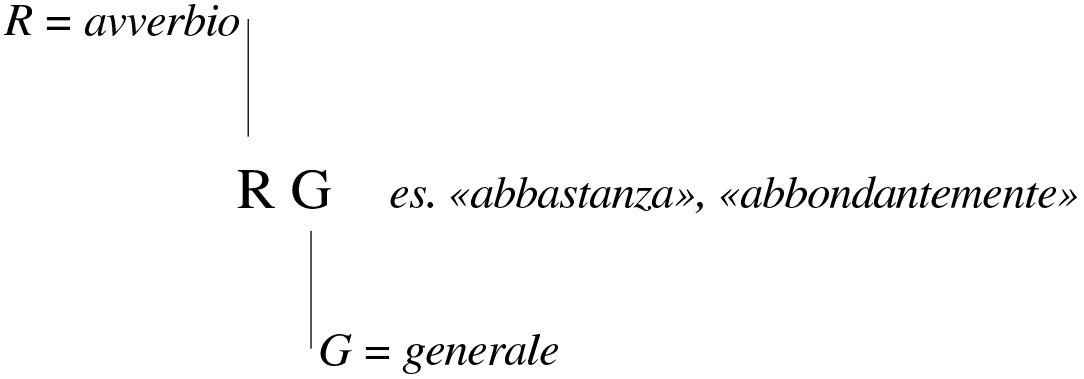
Nella documentazione di Freeling si fa riferimento a un sistema basato sul lavoro di EAGLES (*http://www.ilc.cnr.it/EAGLES96/home.html*), ma questo significa solo che le etichette sarebbero state organizzate per essere conformi alla classificazione prodotta da tale lavoro, dato che uno standard sulla forma di tale etichette non esiste.

La documentazione originale sul sistema di etichettatura usato per la lingua italiana non è stato fornito; tuttavia si osserva che è derivato da quello usato per la lingua spagnola, il quale, invece, ha un'ottima documentazione. Pertanto, gli schemi successivi sono desunti dalla documentazione relativa alla lingua spagnola (*http://nlp.lsi.upc.edu/freeling/doc/tagsets/tagset-es.html*), confrontandone l'utilizzo effettivo nel vocabolario della configurazione italiana.

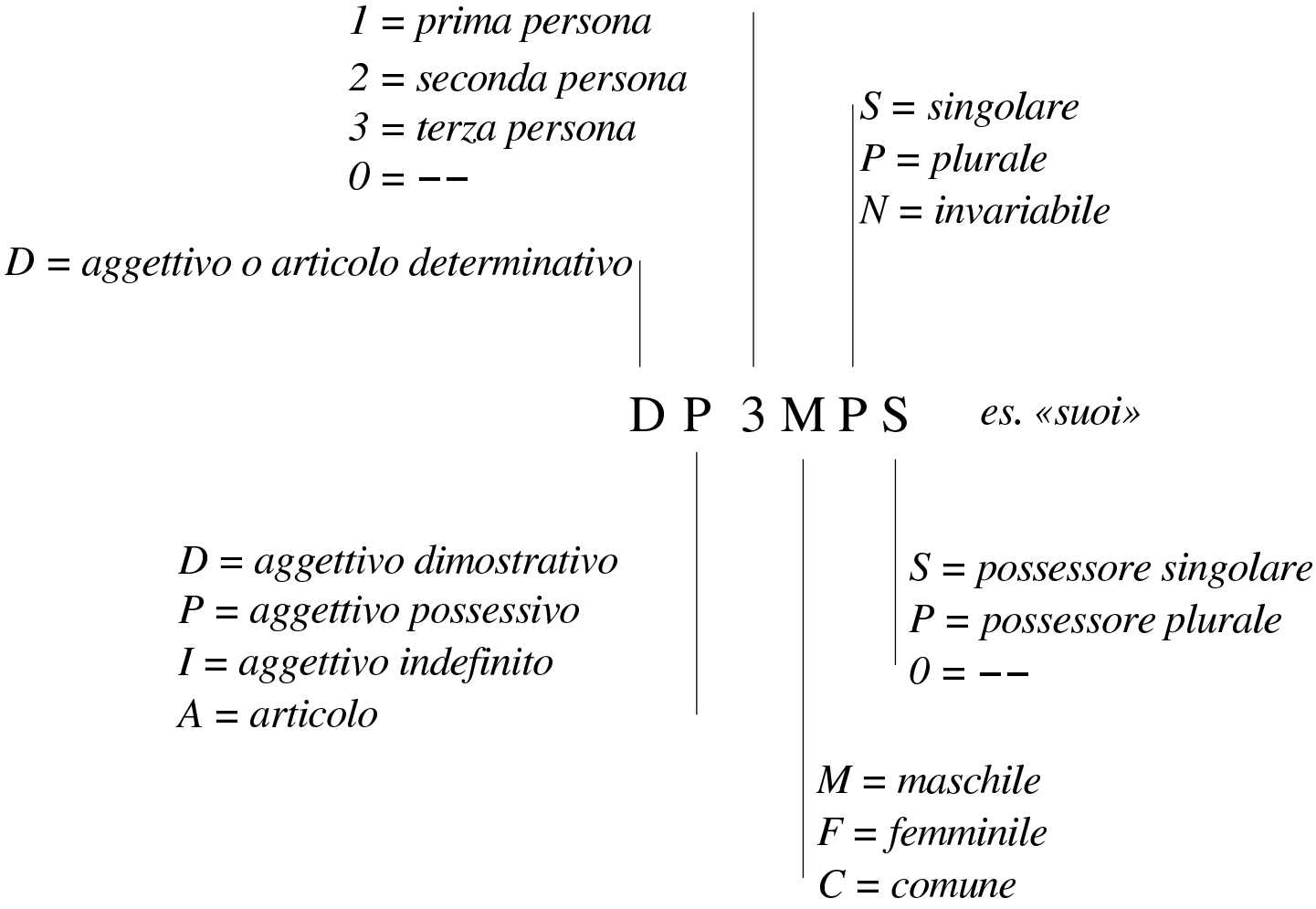
Schema 61.34. Aggettivo.



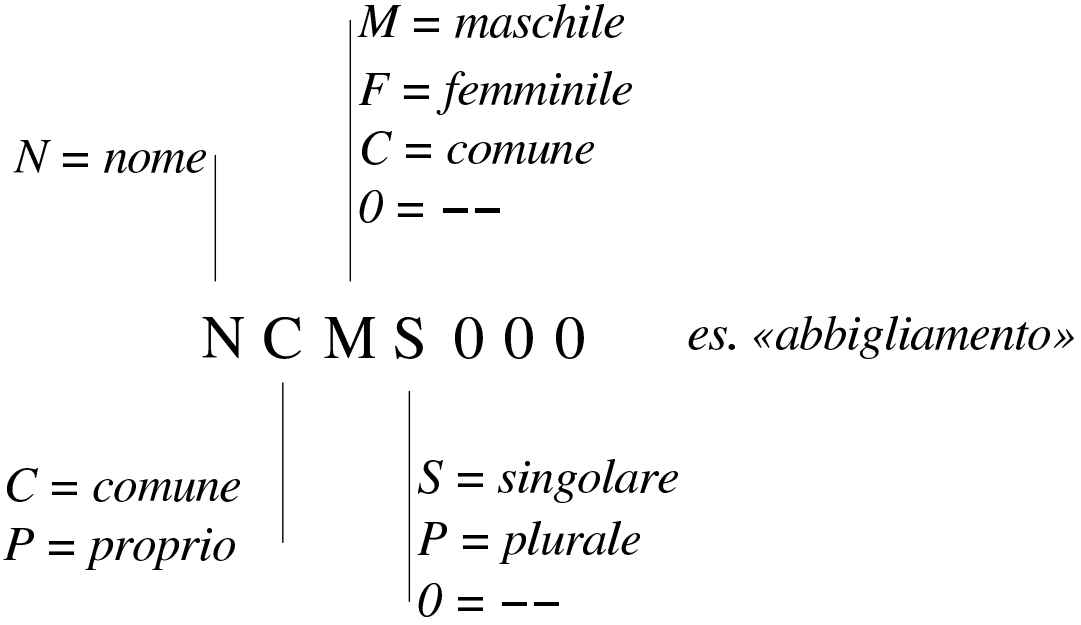
Schema 61.35. Avverbio.



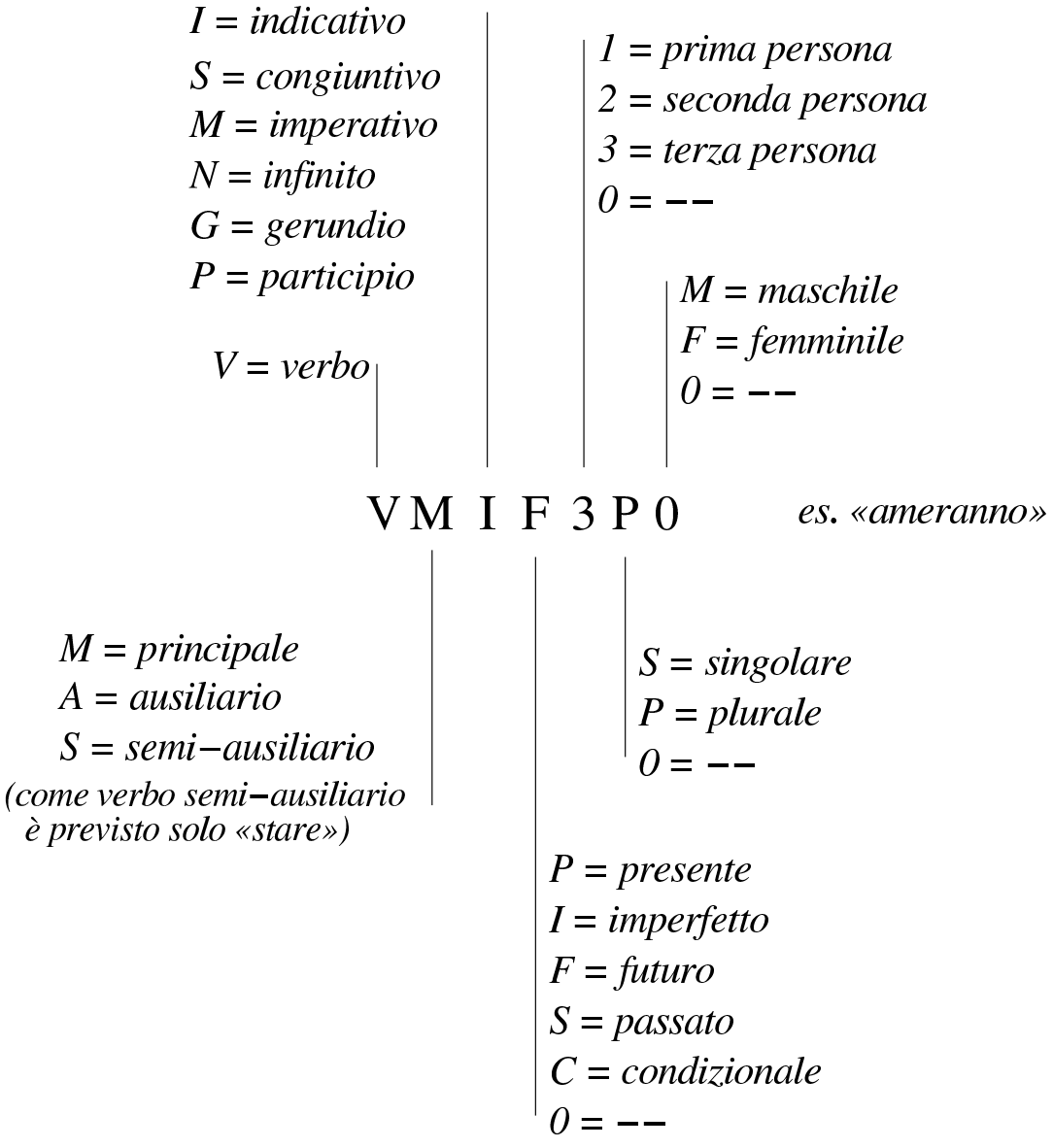
Schema 61.36. Aggettivo o articolo determinativo.



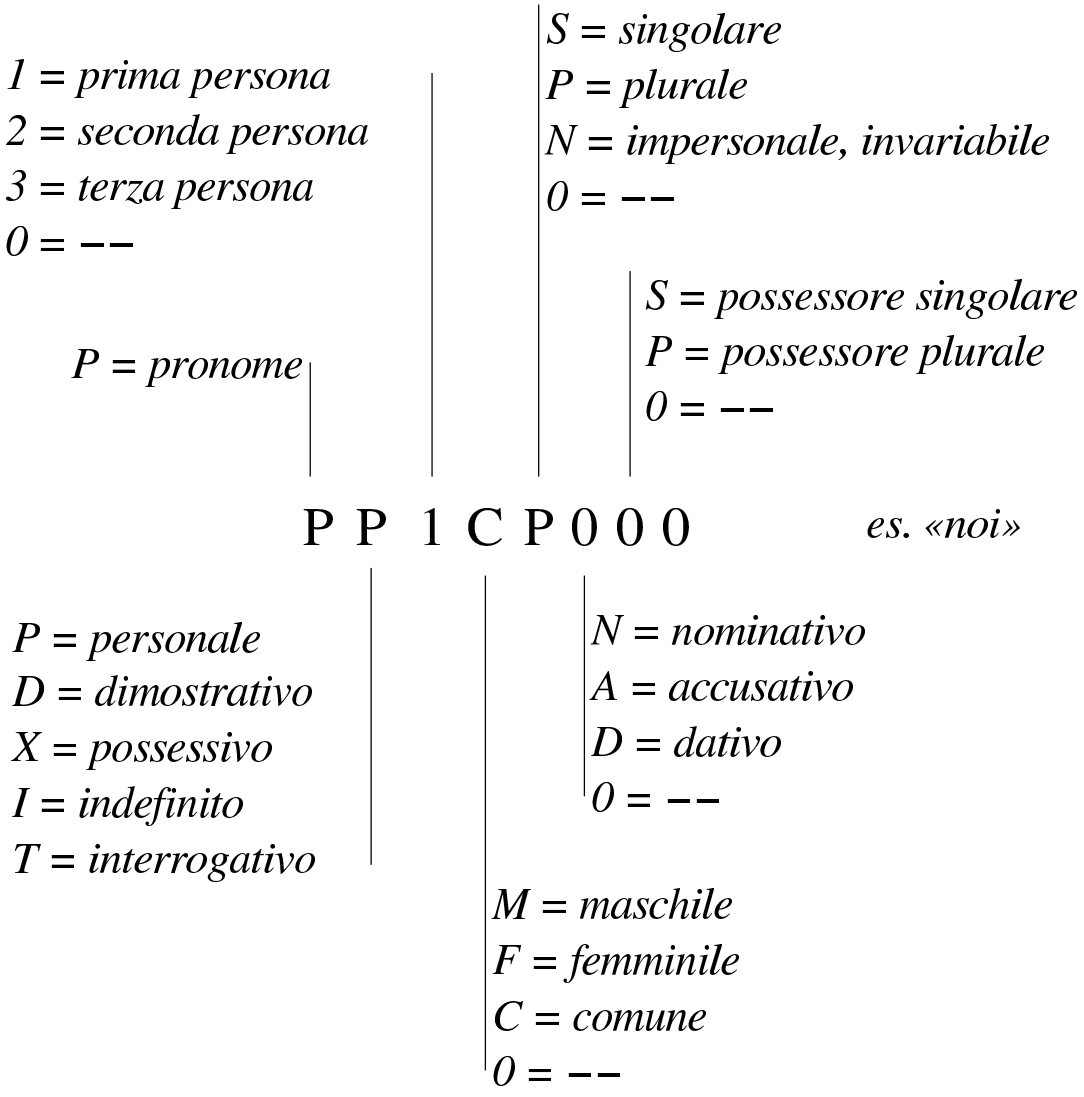
Schema 61.37. Nome.



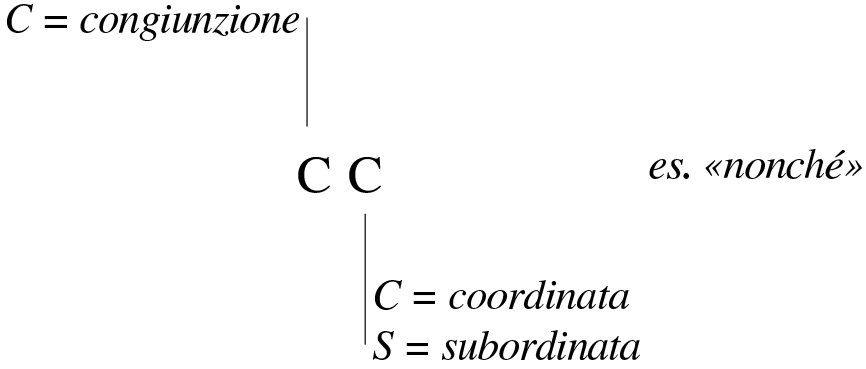
Schema 61.38. Verbo.



Schema 61.39. Pronome.



Schema 61.40. Congiunzione.



Schema 61.41. Interiezione.

I = interiezione

I *es. «ahimé», «alé», «evviva»*

Schema 61.42. Preposizione.

SP = preposizione

M = maschile
F = femminile
0 = ––

S P C F P *es. «nelle», «dalle»*

S = semplice
C = contratta

S = singolare
P = plurale
0 = ––

Schema 61.43. Punteggiatura o parentesi.

F = punteggiatura

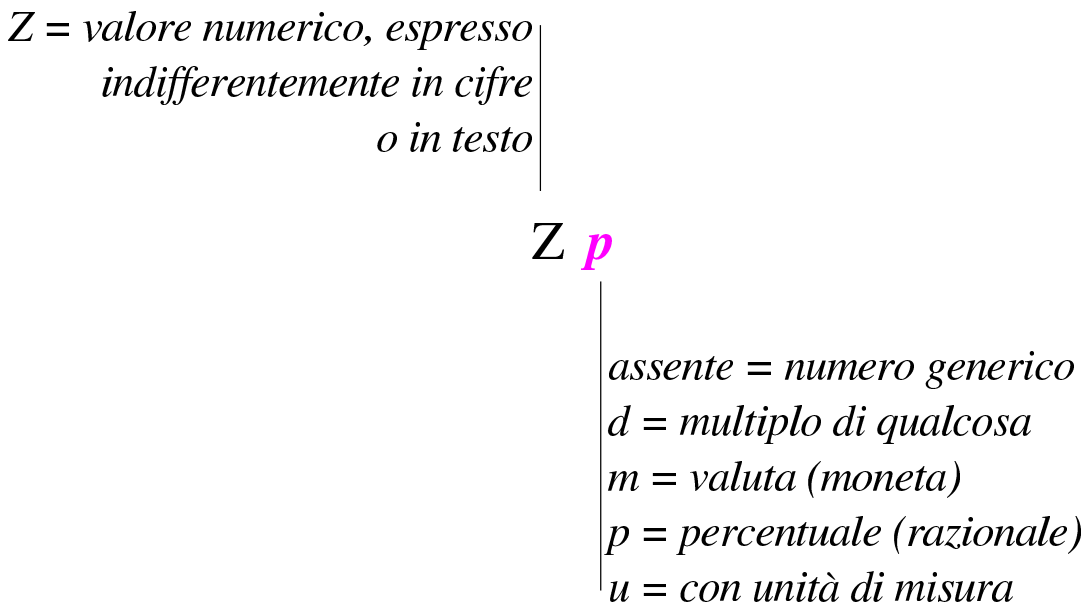
a = apertura
t = chiusura
assente = non applicabile

F r *a*

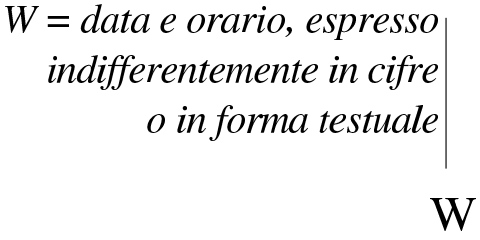
simbolo di punteggiatura, o di parentesi

¡ = Faa	/ = Fh	« = Fra
! = Fat	¿ = Fia	» = Frc
, = Fc	? = Fit	... = Fs
[= Fca	{ = Fla	% = Ft
] = Fct	} = Flt	; = Fx
: = Fd	. = Fp	– = Fz
’ = Fe	(= Fpa	+ = Fz
– = Fg	) = Fpt	= = Fz

Schema 61.44. Dati numerici, in cifre o in lettere.



Schema 61.45. Date e orari.



61.3.2 Uso del sistema già pronto

Freeling è accompagnato da alcuni programmi che consentono di sfruttare il sistema anche se non si costruisce un proprio programma specifico. Ci sono due modalità per lavorare in questo modo: un programma che interroga direttamente i file di dati predisposti e un sistema cliente-servente, con il quale è possibile comunicare anche a distanza, o attraverso altre interfacce.

Per utilizzare questi strumenti è indispensabile specificare il percorso di un file di configurazione, il quale determina la lingua in base alla quale si vuole eseguire un'analisi. Questi file sono collocati normalmente all'interno di '`share/freeling/config/`', ma il per-

corso assoluto effettivo dipende da come esattamente è stato installato Freeling. Per esempio, per la configurazione relativa alla lingua italiana, si deve fare riferimento al file '`it.cfg`'. Questo file di configurazione, in pratica, serve a dare a Freeling tutte le informazioni su dove trovare i file che contengono i dati da utilizzare per l'analisi della lingua selezionata. A ogni modo, tali file si trovano generalmente all'interno di '`share/freeling/`***xx***`/`' (dove ***xx*** rappresenta il linguaggio scelto).

Per l'uso locale e diretto dei file di dati, ci si avvale del programma '**`analyzer`**'; ma per semplicità d'uso, è preferibile servirsi dello script '**`analyze`**' (senza la lettera «r» finale), che fa parte di questi programmi:

```
$ analyze -f /opt/freeling/share/freeling/config/it.cfg ↩
↪      < testo.txt [Invio]
```

L'esempio mostra l'avvio dello script '**`analyze`**', con l'opzione '**`-f`**', attraverso la quale si specifica il percorso del file di configurazione (qui si suppone che Freeling sia stato installato a partire da '`/opt/freeling/`'). Il file '`test.txt`' viene fornito attraverso lo standard input per ottenerne l'analisi. Si suppone che il file contenga il testo seguente:

```
A caval donato, non si guarda in bocca.
```

L'esito dell'analisi sarebbe questa:

```
A a SPS00 1
caval caval NCMN000 0.499985
donato donare VMP00SM 1
non non RG 1
si si PP3CN000 0.5
guarda guardare VMM02S0 0.125
```

```
in in SPS00 1
bocca bocca NCFS000 1
. . Fp 1
```

L'analisi che si ottiene può mostrare più o meno informazioni, a seconda di opzioni particolari. In questo caso si vede, per ogni riga, la parola presa in esame, a fianco della quale appare il lemma (la radice), poi si vede l'etichetta con cui Freeling l'ha classificata e infine la probabilità con cui tale attribuzione è stata fatta (uno equivalente al 100 %). Per esempio, Si vede che la parola «donato» è stata attribuita correttamente al verbo «donare», con etichetta VMP00SM che significa: verbo principale, participio (di cui però non è stato specificato il tempo), singolare, maschile.

Per usare il sistema cliente-servente, occorre avviare prima il servente stabilendo una porta di comunicazione e il file di configurazione:

```
$ analyze 12345 ↩
↪     -f /opt/freeling/share/freeling/config/it.cfg & [Invio]

  Port number '12345' given as first parameter. Starting server mode.

  You can now analyze text with the following command:
    - From this computer:
        analyzer_client localhost 12345 <input.txt >output.txt
    - From any other computer:
        analyzer_client 172.21.11.1 12345 <input.txt >output.txt

  Stop the server with:
        analyze stop 12345
```

Per verificare il funzionamento del servente si può fare una prova con il programma '**`analyzer_client`**':

```
$ analyzer_client localhost 12345 < file.txt [Invio]
```

Eventualmente, ci sono anche esempi di codice PHP che mostrano come comunicare con questo servente.

61.3.3 Altre osservazioni

Freeling è un sistema che può essere riprogrammato per l'analisi della lingua di proprio interesse, eventualmente anche cambiando la simbologia delle etichette scelte. Ma soprattutto ci sono funzioni disponibili solo per alcune lingue e mancanti in italiano: se si è interessati, occorre mettere mano al codice, secondo le indicazioni della documentazione originale.

Va poi osservato che, una volta definito un progetto in cui Freeling debba essere usato, diventa indispensabile scrivere il proprio programma, lasciando da parte quelli predefiniti.

61.3.4 Riferimenti

- *Alicebot*, *http://www.alicebot.org/* , *http://www.alicebot.org/aiml.html* , *http://www.pandorabots.com/pandora/talk?botid=f5d922d97e345aa451*
- Wikipedia, *AIML*, *http://en.wikipedia.org/wiki/AIML*
- Thomas Ringate et al, *AIML Primer*, *http://www.alicebot.org/documentation/aiml-primer.html* , *AIML Reference Manual*, *http://www.alicebot.org/documentation/aiml-reference.html*
- Richard S. Wallace, *Don't Read Me, A.L.I.C.E. and AIML Documentation*, *http://www.alicebot.org/documentation/dont.html* , *Symbolic Reductions in AIML*, *http://www.alicebot.org/documentation/srai.html*

- AI foundation, *Latest release of free ALICE AIML set*, http://code.google.com/p/aiml-en-us-foundation-alice/
- *Program-O AIML Chatbot, an opensource AIML, PHP and MySQL Chatbot*, http://www.program-o.com/ns/ , http://sourceforge.net/projects/program-o/
- *AItools*, http://aitools.org/Main_Page
- *Chatbots*, http://www.chatterbotcollection.com/category_contents.php?id_cat=50 , http://www.chatterbotcollection.com/category_contents.php?id_cat=60
- Bayan Aref Abu Shawar, *A Corpus Based Approach to Generalising a Chatbot System*, http://www.comp.leeds.ac.uk/research/pubs/theses/abushawar.pdf
- John McCarthy, *THE WELL-DESIGNED CHILD*, http://www-formal.stanford.edu/pub/jmc/child/child.html
- Carnegie Mellon University, *CHILDES: Child Language Data Exchange System*, http://childes.psy.cmu.edu/
- The AI training department, Artificial Intelligence NV (AI), *A view from outside the black box*, http://www.a-i.com/data/31-A%20vi20ew%20fr20om%20ou20tside.pdf
- Future Algorithms, *The Corby Home page*, http://futalgo.planetaclix.pt/corby/
- Wikipedia, *Open Mind Common Sense*, http://en.wikipedia.org/wiki/Open_Mind_Common_Sense
- *ConceptNet*, http://conceptnet5.media.mit.edu/

- Luca Serianni e altri, *Italiano*, *II Analisi logica e analisi grammaticale*, punto 2 e punto 4, Garzanti editore, 1997, ISBN 8811504708
- Fabio Tamburini, *La linguistica computazionale: un crogiolo di esperienze multidisciplinari*, http://www.griseldaonline.it/informatica/tamburini.htm
- Wikipedia, *Part-of-speech*, http://en.wikipedia.org/wiki/Part-of-speech , *Part-of-speech tagging*, http://en.wikipedia.org/wiki/Part-of-speech_tagging
- *EAGLES on line*, *Expert Advisory Group on Language Engineering Standards*, http://www.ilc.cnr.it/EAGLES96/home.html
- *Specifiche tecniche per la Treebank sintattico-semantica dell'italiano*, http://www.ilc.cnr.it/tressi_prg/papers/Treebank1_1.pdf
- Wikipedia, *Treebank*, http://en.wikipedia.org/wiki/Treebank
- Wikipedia, *Natural language processing*, http://en.wikipedia.org/wiki/Natural_language_processing
- *Freeling, an open source suite of language analyzers*, http://nlp.lsi.upc.edu/freeling/

[1] *Chatter robot*, *chat bot*, *virtual agent*

[2] **Program-O** GNU GPL

[3] **Freeling** GNU GPL