Introducción a la lingüística computacional con una perspectiva interdisciplinaria.

Jacinto A. Dávila[*,], Luis Astorga[*], Melva Márquez[*,***], Jacobo Myerston[*], M. Marilú Parra[*]**
Universidad de Los Andes, [(*)]Postgrado en Modelado y Simulación,
[(**)]Centro de Investigación y Proyectos en Simulación y Modelos,
[(***)]Escuela de Idiomas Modernos.
Mérida, Venezuela
jacinto@ula.ve, melva@ula.ve

RESUMEN

Este artículo presenta el diseño de un curso introductorio a la lingüística computacional producido por un equipo interdisciplinario conformado por lingüistas y computistas Nuestra hipótesis es que planes de estudios similares pueden contribuir a fortalecer la formación tanto de técnicos como de humanistas para el trabajo de investigación en lingüística general y en ingeniería de contenidos, en particular. Los fundamentos del curso son los ejercicios de lógica computacional. Sin embargo, estos están expresamente dirigidos a soportar las discusiones sobre tópicos específicos y generales de la lingüística.

Palabras claves: Lingüística Computacional, PROLOG.

ABSTRACT

This paper introduces a lecture course on computational linguistics in Spanish. It has been produced by a interdisciplinary team of linguists and computer scientists and it aims to the training of researchers in linguistic, basic and applied, and content engineering. The course is based on a series of exercises from computational logic, specially adapted to serve the analysis of texts in Spanish.

Keywords: Computational lingüistics, PROLOG, Spanish language processing.

INTRODUCCIÒN

Este artículo presenta el diseño de un curso introductorio a la lingüística computacional producido por un equipo interdisciplinario conformado por lingüistas[1] y computistas[2]. La contribución es un plan de estudios que se apoya en herramientas computacionales para mostrar al estudiante conceptos lingüísticos de cara al procesamiento del lenguaje natural. Nuestra hipótesis es que planes de estudios similares pueden contribuir a fortalecer la formación tanto de técnicos como de humanistas para el trabajo de investigación en lingüística general y en ingeniería de contenidos, en particular.

El artículo está organizado de la siguiente manera: En primer lugar, explicamos los objetivos y el plan general del curso que se ha diseñado. A continuación, describimos las herramientas computacionales y los formalismos asociados que usamos en el curso. Seguidamente ilustramos el uso de esas herramientas para abordar un problema troncal de la lingüística en el español: la morfología. En este respecto, destacamos que los programas mostrados fueron diseñados por el equipo interdisciplinario, varios de cuyos participantes no tenían experiencia previa en programación. En la siguiente sección se presentan algunos conceptos de gramática, por entender que esta representa el núcleo duro de la lingüística [6] y que, a nuestro juicio, constituye la base para introducir a los participantes en la descripción de la lengua a través del análisis computacional. Posteriormente, se ilustra la flexibilidad del formalismo computacional, mostrando una adaptación del célebre ejemplo de ELIZA en un programa que reconoce frases en español. La sección siguiente resume

[1] Melva Márquez es profesora de idiomas modernos, con maestría en lingüística aplicada por la U.S.B. y candidata al doctorado en lingüística aplicada por la Universidad Pompeu Fabra de Barcelona, España, Jacobo Myerston es filólogo (*Master of Arts*) y M. Marilú Parra es licenciada en letras y literatura latinoamericana.
[2] Luis Astorga es matemático y Jacinto Dávila es profesor de lógica computacional.

la discusión que el curso plantea en torno a los llamados metalenguajes. La última sección concluye con algunas referencias al plan de investigación que alojará el curso que hemos diseñado.

1.- PRESENTACION DE LA ASIGNATURA

El objetivo general del curso diseñado es conocer tecnologías de procesamiento del lenguaje natural escrito y aplicarlas al español y a las variantes locales de esta lengua. Los integrantes del grupo pretenden que estas tecnologías puedan aplicarse a objetivos tan diversos como estudios de la inteligencia y de la competencia verbal, ejercicios artísticos con el lenguaje, la explotación de recursos informáticos en el ámbito de las ciencias sociales y las humanidades y el modelado de sistemas cognitivos. Así, la asignatura está dirigida a estudiosos de los lenguajes (naturales o artificiales) que hayan tenido alguna exposición mínima a herramientas de computación y que quieran mejorar sus habilidades específicas.

La asignatura tiene los siguientes componentes: fundamentos y tópicos generales en lingüística. Los fundamentos son un recorrido introductorio por el procesamiento del lenguaje natural con PROLOG. La adopción de este lenguaje de programación lógica obedece a: 1) Es un lenguaje declarativo, el programador no está forzado a estructurar sus códigos como un algoritmo con un solo punto de inicio, una secuencia definida de comandos de ejecución y un punto de terminación. Por el contrario, los "textos" en PROLOG son colecciones de cláusulas, reglas que "declaran" conocimiento y que permiten definiciones alternativas del mismo concepto; 2) Es un lenguaje computacional amigable con los no computistas: Los códigos PROLOG están conformados por unos pocos símbolos elementales y mayoritariamente por los símbolos que introduce el programador. Quien codifica no está obligado a respetar una estructura rígida en el código, ni a declarar variables; 3) Es un standard industrial internacional, con realizaciones de bajo costo en todas las plataformas modernas de computación y disponible en forma gratuita para ejercicios académicos; y 4) PROLOG es producto de esfuerzos de investigación, que datan desde comienzos de los años 70 en el siglo XX, tanto en lógica y como en lenguajes formales. Esta tradición incluye varias disciplinas para representación del conocimiento y procesamiento del lenguaje natural.

De hecho, PROLOG fue extendido desde sus inicios con un lenguaje de más alto nivel para la descripción de cierto tipo de gramáticas generativas: *Defined Clause Grammar* [7], [5] o DCG, que presentan formalismos computacionales a los cuales pueden ser fácilmente inducidos los no computistas, en particular los estudiosos de la lingüística. Una regla de producción para una gramática del español, se puede escribir así en DCG:

```
o --> fn, fv.
```

Para indicar que una oración (o) puede ser reducida a una frase sustantival (fn) seguida de una frase verbal (fv). Con esa estructura (la regla de producción), construimos toda una gramática en DCGs:

```
fn --> det, n.
fn --> pron.
fn --> nom_propio.
fv --> v.
fv --> v, fp.
fp --> prop, fn
```

```
det --> [el].
n --> [gato].
pron --> [tu].
prop --> [a].
nom_propio --> [don_quijote].
nom_ propio --> [dulcinea].
```

Ese código puede ser empleado para verificar si cierta oración es gramatical. Por ejemplo, PROLOG entiende la siguiente pregunta:

```
? o ( [don_quijote, ama, a, dulcinea], [ ] ).
```

Respondiendo afirmativamente para indicar que "Don Quijote ama a Dulcinea" es una oración gramatical según las reglas anteriores (que corresponden a un pequeño segmento del Español).

2.- PROCESAMIENTO DEL ESPAÑOL EN PROLOG (EL SILABADOR):

Los programas PROLOG que presentamos y comentamos a continuación intervienen el idioma español en su expresión escrita. El primero [haiku.pl] es específico para verificar la buena construcción métrica de los haikus (forma poética de

origen japonés), y requiere de la ejecución del segundo [silabador.pl], que es un programa de alcance general. Este último descompone las oraciones en su estructura silábica.

No se han considerado las licencias poéticas. Ello requiere un tratamiento especial de la oración en el cual deben compararse sistemáticamente palabras contiguas, así como también debe dar cuenta de las licencias relativas a los hiatos.

2.1.- Haiku.pl

Convierte el verso en una lista de sus sílabas, y verifica con el predicado habitual en Prolog: "append" (en su uso de descomposición) que se cumpla la regla métrica 5-7-5 exigida para un haiku. El programa devuelve 3 variables con las listas de las sílabas de los 3 versos del haiku.

```
% haiku(-Verso_de_5, -Verso_de_7, -Otro_Verso_de_5).
haiku(X,Y,Z) :- silabador(K),  % obtiene el texto del usuario y lo coloca en K como una lista de sílabas.
                append(Primeros,[_,_,_,_,_], K),   % descompone la lista K en Primeros y en otra de 5 silabas.
                 append(X, Y ,Primeros), % Decompone Primeros en X seguida de 7 sílabas.
                 Y =  [_,_,_,_,_,_,_], cinco(X).

cinco([_,_,_,_,_]). %  Una forma alternativa chequear que la lista tiene exactamente 5 elementos.
```

2.2.- silabador.pl

Primero se descompone la oración en una lista de palabras, para luego fragmentar cada una de ellas en sus sílabas constitutivas. La primera operación (getsentence) es un procedimiento para leer el texto desde el teclado del computador y convertirlo en una lista de palabras. Hemos tomado la rutina propuesta por Bratko en [3] (2) .

```
silabador(Silabario) :- getsentence(Wordlist), silaba(Wordlist,Silabario).
```

El programa categoriza los distintos tipos de sílabas según los siguientes criterios:

1) Cuando la sílaba es terminal (incluye las aisladas):
2) La naturaleza de su núcleo vocálico; vocal simple, diptongo o triptongo.
3) Si el núcleo vocálico está precedido por una o dos consonantes como márgenes prenucléicos.
4) La relación del núcleo vocálico de una sílaba con el de la siguiente. Esencial para discriminar la partición de las consonantes entre núcleos vocálicos contiguos según las reglas morfológicas (márgenes pre y posnucléicos).

Comentamos a continuación las secciones del programa, y para ello denotamos a las consonantes por C y a las vocales por A.

1. Se clasifica la sílaba en relación al núcleo vocálico siguiente inmediato o, en su defecto, al vacío que finaliza la palabra. Cada regla lógica "sil(X,Silab)" identifica una construcción silábica posible en nuestro idioma.

```
silaba([],[]).
silaba([P|Resto],Silabario):-   name(P,As),sil(As,Silab),
                                  append(Silab,Silabario1,Silabario),
                                 silaba(Resto,Silabario1).
```

2. Sílabas terminales o aisladas con núcleo vocálico simple:
   V | V-C | C-V-C | C-C-V | C-C-V-C

```
sil([],[]).
```

```
sil([V1|[]],Silab):-vocal(V1),
                    name(Sil,[V1]),
                    append([Sil],Silab1,Silab),
                    sil([],Silab1).

sil([V1,C1|[]],Silab):-vocal(V1),cons(C1),
                       name(Sil,[V1,C1]),
                       append([Sil],Silab1,Silab),
                       sil([],Silab1).

sil([C1,V1|[]],Silab):-cons(C1),vocal(V1),
                       name(Sil,[C1,V1]),
                       append([Sil],Silab1,Silab),
                       sil([],Silab1).

sil([C1,V1,C2|[]],Silab):-cons(C1),vocal(V1),cons(C2),
                          name(Sil,[C1,V1,C2]),
                          append([Sil],Silab1,Silab),
                          sil([],Silab1).

sil([C1,C2,V1|[]],Silab):-cons(C1),cons(C2),vocal(V1),
                          name(Sil,[C1,C2,V1]),
                          append([Sil],Silab1,Silab),
                          sil([],Silab1).

sil([C1,C2,V1,C3|[]],Silab):-cons(C1),cons(C2),vocal(V1),cons(C3),
                             name(Sil,[C1,C2,V1,C3]),
                             append([Sil],Silab1,Silab),
                             sil([],Silab1).
```

3. Sílabas terminales con núcleo vocálico en diptongo o triptongo:
   V-V-C | C-V-V-C | C-C-V-V | C-C-V-V-C | C-V-V-V | C-V-V-V-C

```
sil([C1,V1,V2|[]],Silab):- cons(C1),vocal(V1),vocal(V2),
                           name(Sil,[C1,V1,V2]),
                           append([Sil],Silab1,Silab),
                           sil([],Silab1).

sil([C1,V1,V2,C2|[]],Silab):- cons(C1),vocal(V1),vocal(V2),cons(C2),
                              name(Sil,[C1,V1,V2,C2]),
                              append([Sil],Silab1,Silab),
                              sil([],Silab1).

sil([C1,V1,V2,V3|[]],Silab):- cons(C1),vocal(V1),vocal(V2),vocal(V3),
                              name(Sil,[C1,V1,V2,V3]),
                              append([Sil],Silab1,Silab),
                              sil([],Silab1).

sil([C1,V1,V2,V3,C2|[]],Silab):- cons(C1),vocal(V1),vocal(V2),vocal(V3),cons(C2),
                                 name(Sil,[C1,V1,V2,V3,C2]),
                                 append([Sil],Silab1,Silab),
                                 sil([],Silab1).
```

4. Sílabas no terminales con núcleo vocálico simple:

V | V-C | C-V-C | C-C-V | C-C-V-C | V-C-C | C-V-C-C | C-C-V-C-C

```
sil([V1,C1,V2|Restopalabra],Silab):- vocal(V1),cons(C1),vocal(V2),
                                        Name(Sil,[V1]),
                                       Append([Sil],Silab1,Silab),
                                      sil([C1,V2|Restopalabra],Silab1).

% puesto que la estructura se repite, hemos suprimido algunas clausulas para ahorrar espacio.

sil([C1,V1,C2,C3,V2|Restopalabra],Silab):- cons(C1),vocal(V1),cons(C2),cons(C3),vocal(V2),
                                             name(Sil,[C1,V1,C2]),
                                             append([Sil],Silab1,Silab),
                                            sil([C3,V2|Restopalabra],Silab1).

sil([C1,C2,V1,C3,C4,V2|Restopalabra],Silab):-cons(C1),cons(C2),vocal(V1),
                                              cons(C3),cons(C4), vocal(V2),
                                                cluster(C3,C4),
                                                 name(Sil,[C1,C2,V1]),
                                                 append([Sil],Silab1,Silab),
                                                sil([C3,C4,V2|Restopalabra],Silab1).

sil([C1,C2,V1,C3,C4,V2|Restopalabra],Silab):-cons(C1),cons(C2),vocal(V1),
                                              cons(C3),cons(C4), vocal(V2),
                                                C3=C4,
                                                 name(Sil,[C1,C2,V1]),
                                                 append([Sil],Silab1,Silab),
                                                sil([C3,C4,V2|Restopalabra],Silab1).

sil([C1,C2,V1,C3,C4,V2|Restopalabra],Silab):-cons(C1),cons(C2),vocal(V1),
                                              cons(C3),cons(C4), vocal(V2),
                                                 name(Sil,[C1,C2,V1,C3]),
                                                 append([Sil],Silab1,Silab),
                                                sil([C4,V2|Restopalabra],Silab1).

sil([V1,C1,C2,C3,V2|Restopalabra],Silab):- vocal(V1),cons(C1),cons(C2),
                                           cons(C3),vocal(V2),
                                            C2=115,
                                             name(Sil,[V1,C1,C2]),
                                             append([Sil],Silab1,Silab),
                                            sil([C3,V2|Restopalabra],Silab1).

sil([V1,C1,C2,C3,V2|Restopalabra],Silab):- vocal(V1),cons(C1),cons(C2),cons(C3),vocal(V2),
                                             name(Sil,[V1,C1]),
                                             append([Sil],Silab1,Silab),
                                            sil([C2,C3,V2|Restopalabra],Silab1).

sil([C1,V1,C2,C3,C4,V2|Restopalabra],Silab):- cons(C1),vocal(V1),cons(C2),
                                               cons(C3),cons(C4), vocal(V2),
                                                 C3=115,
                                                  name(Sil,[C1,V1,C2,C3]),
                                                  append([Sil],Silab1,Silab),
                                                 sil([C4,V2|Restopalabra],Silab1).

sil([C1,V1,C2,C3,C4,V2|Restopalabra],Silab):-cons(C1),vocal(V1),cons(C2),
                                              cons(C3),cons(C4),vocal(V2),
                                                 name(Sil,[C1,V1,C2]),
```

```
                                                append([Sil],Silab1,Silab),
                                               sil([C3,C4,V2|Restopalabra],Silab1).

sil([C1,C2,V1,C3,C4,C5,V2|Restopalabra],Silab):- cons(C1),cons(C2),vocal(V1),cons(C3),
                                                  cons(C4),cons(C5),vocal(V2),
                                                    C4=115,
                                                     name(Sil,[C1,C2,V1,C3,C4]),
                                                     append([Sil],Silab1,Silab),
                                                    sil([C5,V2|Restopalabra],Silab1).

sil([C1,C2,V1,C3,C4,C5,V2|Restopalabra],Silab):- cons(C1),cons(C2),vocal(V1),cons(C3),
                                                  cons(C4), cons(C5),vocal(V2),
                                                     name(Sil,[C1,C2,V1,C3]),
                                                     append([Sil],Silab1,Silab),
                                                    sil([C4,C5,V2|Restopalabra],Silab1).

sil([V1,C1,C2,C3,C4,V2|Restopalabra],Silab):- vocal(V1),cons(C1),cons(C2),
                                              cons(C3),cons(C4), vocal(V2),
                                                 C2=115,
                                                  name(Sil,[V1,C1,C2]),
                                                 append([Sil],Silab1,Silab),
                                                sil([C3,C4,V2|Restopalabra],Silab1).

sil([C1,V1,C2,C3,C4,C5,V2|Restopalabra],Silab):- cons(C1),vocal(V1),cons(C2),cons(C3),
                                                  cons(C4), cons(C5),vocal(V2),
                                                    C3=115,
                                                     name(Sil,[C1,V1,C2,C3]),
                                                     append([Sil],Silab1,Silab),
                                                    sil([C4,C5,V2|Restopalabra],Silab1).

sil([C1,C2,V1,C3,C4,C5,C6,V2|Restopalabra],Silab):-cons(C1),cons(C2),vocal(V1),cons(C3),
                                                      cons(C4),cons(C5),cons(C6),vocal(V2),
                                                    C4=115,
                                                     name(Sil,[C1,C2,V1,C3,C4]),
                                                     append([Sil],Silab1,Silab),
                                                    sil([C5,C6,V2|Restopalabra],Silab1).
```

5. Sílabas no terminales con diptongo :
   V-V | V-V-C | C-V-V-C | C-C-V-V | C-C-V-V-C | V-V-C-C | C-V-V-C-C |
   C-C-V-V-C-C

```
sil([V1,V2,C1,V3|Restopalabra],Silab):- vocal(V1),vocal(V2),cons(C1),vocal(V3),
                                        Name(Sil,[V1,V2]),
                                        Append([Sil],Silab1,Silab),
                                          sil([C1,V3|Restopalabra],Silab1).

sil([C1,V1,V2,C2,V3|Restopalabra],Silab):- cons(C1),vocal(V1),vocal(V2),cons(C2),vocal(V3),
                                           name(Sil,[C1,V1,V2]),
                                           append([Sil],Silab1,Silab),
                                           sil([C2,V3|Restopalabra],Silab1).

sil([C1,C2,V1,V2,C3,V3|Restopalabra],Silab):- cons(C1),cons(C2),vocal(V1),
                                              vocal(V2),
                                                 cons(C3),vocal(V3),
                                                 name(Sil,[C1,C2,V1,V2]),
```

```
                                                append([Sil],Silab1,Silab),
                                               sil([C3,V3|Restopalabra],Silab1).

sil([V1,V2,C1,C2,V3|Restopalabra],Silab):- vocal(V1),vocal(V2),cons(C1),cons(C2),vocal(V3),
                                             cluster(C1,C2),
                                              name(Sil,[V1,V2]),
                                              append([Sil],Silab1,Silab),
                                             sil([C1,C2,V3|Restopalabra],Silab1).

sil([V1,V2,C1,C2,V3|Restopalabra],Silab):- vocal(V1),vocal(V2),cons(C1),cons(C2),vocal(V3),
                                             C1=C2,
                                              name(Sil,[V1,V2]),
                                              append([Sil],Silab1,Silab),
                                             sil([C1,C2,V3|Restopalabra],Silab1).

sil([V1,V2,C1,C2,V3|Restopalabra],Silab):- vocal(V1),vocal(V2),cons(C1),cons(C2),vocal(V3),
                                              name(Sil,[V1,V2,C1]),
                                       append([Sil],Silab1,Silab),
                                              sil([C2,V3|Restopalabra],Silab1).

sil([C1,V1,V2,C2,C3,V3|Restopalabra],Silab):- cons(C1),vocal(V1),vocal(V2),
                                               cons(C2), cons(C3),vocal(V3),
                                                 cluster(C2,C3),
                                                  name(Sil,[C1,V1,V2]),
                                                  append([Sil],Silab1,Silab),
                                                 sil([C2,C3,V3|Restopalabra],Silab1).

sil([C1,V1,V2,C2,C3,V3|Restopalabra],Silab):- cons(C1),vocal(V1),vocal(V2),
                                               cons(C2), cons(C3),vocal(V3),
                                                  C2=C3,
                                                   name(Sil,[C1,V1,V2]),
                                                   append([Sil],Silab1,Silab),
                                                  sil([C2,C3,V3|Restopalabra],Silab1).

sil([C1,V1,V2,C2,C3,V3|Restopalabra],Silab):- cons(C1),vocal(V1),vocal(V2),
                                               cons(C2),cons(C3),vocal(V3),
                                                  name(Sil,[C1,V1,V2,C2]),
                                                  append([Sil],Silab1,Silab),
                                                 sil([C3,V3|Restopalabra],Silab1).

sil([C1,C2,V1,V2,C3,C4,V3|Restopalabra],Silab):- cons(C1),cons(C2),vocal(V1),vocal(V2),
                                                    cons(C3),cons(C4),vocal(V3),
                                                  cluster(C3,C4),
                                                   name(Sil,[C1,C2,V1,V2]),
                                                   append([Sil],Silab1,Silab),
                                                  sil([C3,C4,V3|Restopalabra],Silab1).

sil([C1,C2,V1,V2,C3,C4,V3|Restopalabra],Silab):- cons(C1),cons(C2),vocal(V1),vocal(V2),
                                                   cons(C3),cons(C4),vocal(V3),
                                                   C3=C4,
                                                   name(Sil,[C1,C2,V1,V2]),
                                                   append([Sil],Silab1,Silab),
                                                 sil([C3,C4,V3|Restopalabra],Silab1).

sil([C1,C2,V1,V2,C3,C4,V3|Restopalabra],Silab):-cons(C1),cons(C2),vocal(V1),vocal(V2),
                                                   cons(C3),cons(C4),vocal(V3),
                                                   name(Sil,[C1,C2,V1,V2,C3]),
```

```
                                                    append([Sil],Silab1,Silab),
                                                   sil([C4,V3|Restopalabra],Silab1).

sil([V1,V2,C1,C2,C3,V3|Restopalabra],Silab):- vocal(V1),vocal(V2),cons(C1),cons(C2),
                                                    cons(C3),vocal(V3),
                                                   C2=115,
                                                    name(Sil,[V1,V2,C1,C2]),
                                                   append([Sil],Silab1,Silab),
                                                  sil([C3,V3|Restopalabra],Silab1).

sil([V1,V2,C1,C2,C3,V2|Restopalabra],Silab):- vocal(V1),vocal(V2),cons(C1),cons(C2),
                                                    cons(C3),vocal(V3),
                                                    name(Sil,[V1,V2,C1]),
                                                    append([Sil],Silab1,Silab),
                                                  sil([C2,C3,V3|Restopalabra],Silab1).

sil([C1,V1,V2,C2,C3,C4,V3|Restopalabra],Silab):- cons(C1),vocal(V1),vocal(V2),cons(C2),
                                                     cons(C3),cons(C4),vocal(V3),
                                                    C3=115,
                                                     name(Sil,[C1,V1,V2,C2,C3]),
                                                     append([Sil],Silab1,Silab),
                                                    sil([C4,V3|Restopalabra],Silab1).

sil([C1,V1,V2,C2,C3,C4,V3|Restopalabra],Silab):-cons(C1),vocal(V1),vocal(V2),cons(C2),
                                                    cons(C3),cons(C4),vocal(V3),
                                                    name(Sil,[C1,V1,V2,C2]),
                                                    append([Sil],Silab1,Silab),
                                                   sil([C3,C4,V3|Restopalabra],Silab1).

sil([C1,C2,V1,V2,C3,C4,C5,V3|Restopalabra],Silab):- cons(C1),cons(C2),vocal(V1),vocal(V2),
                                                      cons(C3),cons(C4),cons(C5),vocal(V3),
                                                        C4=115,
                                                         name(Sil,[C1,C2,V1,V2,C3,C4]),
                                                        append([Sil],Silab1,Silab),
                                                       sil([C5,V3|Restopalabra],Silab1).

sil([C1,C2,V1,V2,C3,C4,C5,V3|Restopalabra],Silab):-cons(C1),cons(C2),vocal(V1),vocal(V2),
                                                     cons(C3),cons(C4),cons(C5),vocal(V3),
                                                        name(Sil,[C1,C2,V1,V2,C3]),
                                                        append([Sil],Silab1,Silab),
                                                      sil([C4,C5,V3|Restopalabra],Silab1).

sil([V1,V2,C1,C2,C3,C4,V3|Restopalabra],Silab):- vocal(V1),vocal(V2),cons(C1),cons(C2),
                                                    cons(C3),cons(C4),vocal(V3),
                                                   C2=115,
                                                    name(Sil,[V1,V2,C1,C2]),
                                                   append([Sil],Silab1,Silab),
                                                  sil([C3,C4,V3|Restopalabra],Silab1).


sil([C1,V1,V2,C2,C3,C4,C5,V3|Restopalabra],Silab):- cons(C1),vocal(V1),vocal(V2),cons(C2),
                                                      cons(C3),cons(C4),cons(C5),vocal(V3),
                                                     C3=115,
                                                      name(Sil,[C1,V1,V2,C2,C3]),
                                                      append([Sil],Silab1,Silab),
                                                     sil([C4,C5,V3|Restopalabra],Silab1).
```

```
sil([C1,C2,V1,V2,C3,C4,C5,C6,V3|Restopalabra],Silab):- cons(C1),cons(C2),vocal(V1),
                                                       vocal(V2), cons(C3), cons(C4),
                                                       cons(C5),cons(C6),vocal(V3),
                                                         C4=115,
                                                         name(Sil,[C1,C2,V1,V2,C3,C4]),
                                                         append([Sil],Silab1,Silab),
                                                        sil([C5,C6,V3|Restopalabra],Silab1).
```

6. Sílabas no terminales con triptongo : C-V-V-V

```
sil([C1,V1,V2,V3,C2,V4|Restopalabra],Silab):- cons(C1),vocal(V1),vocal(V2),vocal(V3),
                                              cons(C2),vocal(V4),
                                                 name(Sil,[C1,V1,V2,V3]),
                                                append([Sil],Silab1,Silab),
                                               sil([C2,V4|Restopalabra],Silab1).
```

7. Se verifica la naturaleza de las letras (consonantes o vocales)

```
% Los números corresponden a la codificación de las letras del español, e.g. el 241 es la ñ.
Cons(C):-  member(C, [122,121,120,119,118,116,115,114,113,112,110]).
Cons(C):-  member(C,[109, 241,108,107,106,104,103,102,100,99,98]).
Vocal(V):- member(V, [121,117,111,105,101,97] ).
```

8. Se controla la combinación de los fonemas oclusivos / p, t, k / , los sonoros / b, d, g / y el fricativo / f / con las líquidas / l, r / [1].

```
cluster(C1, C2) :- member(C1, [112,116,99,107,102,98,100,103,110]),
                   member(C2, [108,114,104]).
```

La siguiente figura muestra el funcionamiento del programa:

```
% c:/cursos/linguistica_computacional_a2001/haikus.pl compiled 0.00 sec, 24,360 bytes

?- haiku(Verso5, Verso7, OVerso5 ).
|: cada comarca tiene los fanatismos que se merece.


Verso5 = [ca, da, co, mar, ca]
Verso7 = [tie, ne, los, fa, na, tis, mos]
OVerso5 = [que, se, me, re, ce]

Yes
```

3.- INTRODUCCION A LAS GRAMÀTICAS COMTEMPORANEAS

En este módulo de la asignatura se ofrece al participante del curso la exposición de conceptos fundamentales como la gramaticalidad [9] y la subcategorización verbal, así como una panorámica de las principales corrientes gramaticales que hoy día existen para describir y modelar el procesamiento del lenguaje natural, tales como la Gramática Léxico Funcional (LFG), la Gramática de Estructura de Frase Generalizada (GPSG) y en especial, la Gramática de Estructura de Frase Regida por el Núcleo (HPSG). Este último modelo sintáctico y lexicalista toma de las otras dos gramáticas

anteriores aspectos importantes para la descripción del lenguaje [8]; propone, además, la representación de las unidades lingüísticas como entradas léxicas en estructuras de rasgos de par atributo y valor que coindexan información léxica, sintáctica, semántica y pragmática. Con ello se pretende iniciar al participante en el reconocimiento de los formalismos lingüísticos utilizados en la lingüística moderna, de manera que pueda ser capaz de profundizarse en su estudio si decide tomar esta línea de investigación en su tesis de grado.

4.- ELIZAUL: UNA INTRODUCCIÒN A LA SOCIOLINGUISTICA Y A LA PRAGMÀTICA

La relación del significado del signo con el contexto ha sido y sigue siendo uno de los problemas más complejos que ha enfrentando la lingüística y las disciplinas hermenéuticas en general. El problema consiste en describir el lenguaje como un fenómeno social que no tiene existencia independiente del hablante; se trata de desobjetivar la comunicación y reconocer el hecho intencional sobre el cual subyace. El proceso del descubrimiento del contexto ha sido lento en la larga historia de la cultura occidental: alrededor de unos 2500 años. Se ha pasado de la visión etimológica –de la búsqueda del significado original de las palabras – al concepto del *uso lingüístico* introducido por Ludwig Wittgenstein en sus *Investigaciones Filosóficas* [12]. En la antigüedad griega, los filósofos pensaban que las sonidos que constituyen las palabras están íntimamente ligados a la esencia de la cosa misma, es decir que el significado es absoluto e independiente de quien lo utilice.

Hoy en día nuestra visión de la cosas ha cambiado, ya no esperamos entender el mundo a través de la morfología y fonética de las palabras. Hoy en día preferimos entender al ser humano a través del uso que hace éste del lenguaje y de los símbolos en general. Hemos descubierto también, gracias a Austin [2] que con el lenguaje no sólo decimos cosas, sino que hacemos algo: hablando construimos nuestra identidad, cambiamos de status, nos comprometemos y hasta somos capaces de inventar mundos que aún no conocemos. El uso que hacemos del lenguaje nos permite indicar a qué grupo social pertenecemos, qué tipo de persona somos, en qué situación nos encontramos y de qué manera se debe interpretar lo que estamos diciendo. A cada situación le corresponde un tipo particular de vocabulario y un cierto tono de voz. No se trata solamente del uso especializado del lenguaje, sino de los marcos que constituyen nuestra experiencia[3].

Un ejemplo muy ilustrativo de lo que es el *marco lingüístico* se puede observar en la ya celebre Eliza, una pieza de software basada en plantillas que pretende simular una sesión psicoanalítica. Las plantillas son sensibles a ciertas palabras claves que invocan una frase previamente diseñada: si el usuario menciona la palabra *mother*, Eliza le responderá: *tell me more about your family*. Eliza está lejos de ser un software inteligente, si por inteligente se entiende la capacidad de comprender a nivel semántico los enunciados del diálogo, pero se adecua muy bien para el modelado y simulación de diálogos en un contexto determinado.

La coherencia de una conversación con un software de este tipo depende de que el usuario identifique correctamente el contexto y el papel que debe desempeñar en éste. En casos en que se ha intentado tomarle el pelo a Eliza, rompiendo con el marco contextual, se han obtenido frecuentemente respuestas incoherentes.

Durante el diseño de nuestro curso se modificó una versión de Eliza hecha en Prolog y se adaptó a otro contexto. A la nueva versión, que se comporta como un psicoanalista un poco atípico, la hemos llamado Elizaul, El marco de la consulta psicológica se ha mantenido, pero se le ha introducido un léxico que no corresponde a la situación. Nuestro analista padece un trastorno del lenguaje llamado *coprolalia* que se caracteriza por un habla obscena compulsiva que, en condiciones normales, podría resultar chocante y que haría pensar al paciente necesitado de ayuda que su terapeuta está loco. En este ejercicio no se trata de simular una situación excéntrica, sino más bien de explorar los marcos contextuales a través de una transgresión. Esta técnica, que consiste en poner léxico propio de contextos sociales de poco prestigio en la boca de actores “respetables”, ha sido utilizada en diversas tradiciones artísticas y tiene como objetivo demarcar los límites de los marcos lingüísticos y socioculturales por medio del contraste y la descontextualización .

Ejercicios similares pueden servir para explorar, simular y describir un gran número de *actos performativos* en los que por lo menos uno de los participantes cambia de status al finalizar el mismo. En el caso de Eliza se espera que el diálogo lleve al alivio del paciente y que éste pase del estatus de enfermo al de sano. Elizaul, por otra parte, intenta transformar al interlocutor a través de la risa, llevándolo de un estado de marcadas restricciones sociales a uno de menor control social[4]. De la misma manera se podría usar esta técnica para el estudio de la confesión católica o de la consulta médica como actos performativos, tratando de capturar el léxico propio del contexto, los actores y sus patrones de interacción, para luego someterlos a un proceso de modelado que puede reproducir de manera verosímil un situación sociolingüística.

---

[3] Utilizamos el término marco en el sentido que le dio Erwin Goffman en su libro Frame-Analysis, an essay on the Organization of Experience, New York, 1974

[4] En este sentido ver Mary Douglas, Social Control of Cognition; Factors in Joke Perception, en Man, N. S. 3, 3, pp. 361-367

5.- METALENGUAJES

Llamamos metalenguajes a las especificaciones para *manipular el conocimiento racional y lingüístico*, es decir, un conjunto de convenciones que norman la comunicación en un lenguaje. El metalenguaje es un lenguaje que trata sobre lenguaje(s). En la sección de los metalenguajes se le ofrece al participante una visión en conjunto de los estándares internacionales utilizados para representar textos en formato electrónico. Los lenguajes de marcaje que se introducen son el SGML (*Standard Generalized Markup Language*) y el XML (*Extended Markup Language*). En el caso del lenguaje SGML se introduce la noción de tipo de documento y Definición de Tipo de Documento (DTD), las cuales se utilizan para dar formato a documentos existentes escritos en SGML. Uno de los principales objetivos planteados bajo esta sección es la vinculación de los recursos informáticos con **el procesamiento de corpora lingüísticos**, de manera que de su funcionamiento no sólo puedan beneficiarse las investigaciones de lingüistas y computistas, sino todas aquellas pesquisas que de una u otra manera estén relacionadas con la descripción del lenguaje natural.

A continuación, un ejemplo de los documentos XML que se estudian en el curso:

```
<?xml version="1.0"?>
<!--Nombre del archivo: gramaticaprolog.xml -->
<?xml-stylesheet type="text/css" href="gramaticaprolog.css"?>
<oracion>
<fn>
<nombre>
<n1>Don Quijote</n1>
</nombre>
<rasgosdeunificacion>
<genero>masculino</genero>
<numero>singular</numero>
</rasgosdeunificacion>
</fn>
<fv>
<verbo>ama
<tiempo>presente</tiempo>
<persona>tercera</persona>
<numero>singular</numero>
</verbo>
<fp>
<prep>a</prep>
<fn>
<objetodir>Dulcinea</objetodir>
</fn>
</fp>
</fv>
</oracion>
```

CONCLUSIONES

Hemos diseñado un curso para inducir a computistas y a lingüistas a compartir sus conocimientos y a aprovechar las tecnologías de la información para el estudio de los lenguajes naturales. Los fundamentos del curso son los ejercicios de lógica computacional. Sin embargo, estos están expresamente dirigidos a soportar las discusiones sobre tópicos específicos y generales de la lingüística.

Creemos que el estudio de los lenguajes debe ocurrir en un entorno en el que, a la par del rigor científico, se entienda que el lenguaje es y será “constantemente innovador, ilimitado, libre, al parecer, del control de estímulos externos o estados de ánimos internos, coherente y apropiado a las situaciones” [4]. Por ellos hemos hecho un esfuerzo para realizar este ejercicio interdisciplinario.

REFERENCIAS

1. Alarcos Llorach, E. : *Gramática de la Lengua Española*, Espasa-Calpe. Madrid, 1994.
2. Austin, J.L. How to do things with words, Oxford University Press, 1962.
3. Bratko, I.: *Prolog, Programming for Artificial Intelligence* (2da ed), Addison- Wesley. Harlow, 1990.
4. Chomsky, N. *El lenguaje y los problemas del conocimiento*. Madrid: Visor.
5. Covington, Michael. Natural Language Processing for Prolog Programmers. Prentice Hall. Englewood Cliffs. NJ, 07632. 1994. ISBN 0-13-629213-5
6. Lerat, P. (1997)(Trad. Albert Ribas) Las lenguas especializadas. Barcelona: Ariel (título original: Les langues spécialisées, 1995)
7. Pereira, F. and Sheiber, S. "Prolog and Natural-Language Analysis," Center for the Study of Language and Information, 1987.
8. Pollard C., y Sag, I. Sag *Head-Driven Phrase Structure Grammar*. Stanford, CA: CSLI Lecture Note Series, Center for the Study of Languga and Information. (1993*)*
9. Sells, P. (1985) *Lectures on Contemporary Syntactic Theories*. Stanford, CA: Center for the Study of Language and Information.
10. Tuson, J. (Dir.) (2000) *Diccionari de Lingüística*. Barcelona: Bibliograf.
11. Von Kloop, Ana. A Practical Introduction to Prolog and Computacional Linguistics. Lecture notes. Disponibles desde email://a.von.klopp@bangor.ac.uk.
12. Luwig Wittgenstein, Philosophische Untersuchungen, Werkausgabe, Frankfurt am Main, Surhkamp 1984, pp. 255-581