UNA PROPUESTA DE PRÁCTICAS DE LENGUAJE NATURAL

Antonio Molina, Lidia Moreno

Facultad de Informática de la Universidad Politécnica de Valencia
e-mail: {amolina,lmoreno}@dsic.upv.es

Resumen: En este artículo se presentan las prácticas de la asignatura Lenguaje Natural que se imparte en el título de Ingeniero en Informática de la Universidad Politécnica de Valencia. Éstas consisten en la implementación de un sistema básico de procesamiento de lenguaje natural (análisis léxico, sintáctico e interpretación semántica) que se integra en un entorno de interrogación a una base de datos.

1.- INTRODUCCIÓN

El Procesamiento del Lenguaje Natural (PLN) es una parte esencial de la Inteligencia Artificial, que utiliza las técnicas de análisis y de representación del conocimiento que esta provee. Además, el PLN está directamente relacionado con otras materias como la Lingüística, ciencia que estudia la formalización del lenguaje, la Psicolingüística, que estudia los mecanismos humanos de comprensión, la Teoría Formal de Lenguajes, que proporciona técnicas de análisis, o la Lógica como mecanismo de representación del significado. El programa teórico de la asignatura Lenguaje Natural[1] [Moreno98] aborda desde diversas aproximaciones las sucesivas etapas de análisis que sufre una oración, que se implementarán en el marco del programa práctico.

2.- OBJETIVOS

El objetivo fundamental del programa práctico es que el alumno aplique los conocimientos adquiridos en las clases teóricas para abordar las fases de análisis de un sistema de PLN. Todo esto se llevará a cabo mediante la

[1] Lenguaje Natural es una asignatura optativa que se imparte en 5° curso de la Facultad de Informática de la UPV dentro de la intensificación de Inteligencia Artificial.

programación lógica. De esta forma, al finalizar las prácticas el alumno será capaz de abordar la implementación de un sistema de PLN, al menos en las fases iniciales de análisis (Léxico, Sintáctico y Semántico). Evidentemente, el sistema implementado será muy restringido (en cuanto al vocabulario reconocido, las estructuras sintácticas permitidas y la complejidad semántica de las oraciones) pero los mecanismos que el alumno desarrolla (definición de entradas léxicas, reglas gramaticales o el mecanismo de composición de formas lógicas) son perfectamente generalizables a sistemas de mayor complejidad.

3.- MOTIVACIÓN

Uno de los aspectos fundamentales en el proceso enseñanza-aprendizaje es la motivación del alumno. Es esencial despertar en el alumno el interés por la asignatura, enumerando las distintas aplicaciones del PLN: sistemas flexibles de acceso a bases de datos, sistemas de extracción de información, interacción con sistemas expertos, traducción automática, etc.

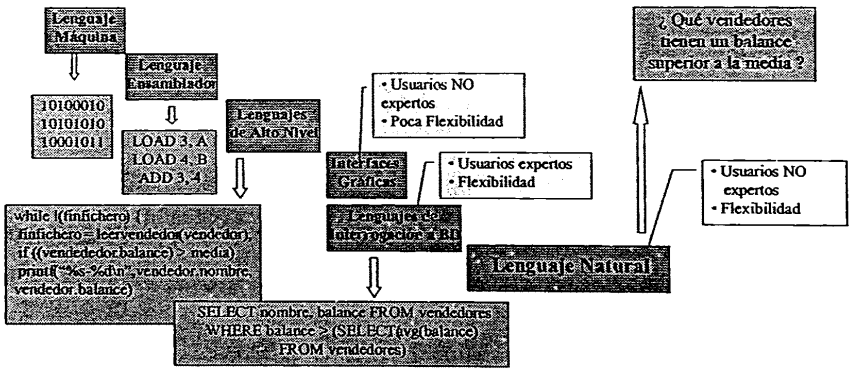


Figura 1: Evolución en la interfaz hombre-máquina para el acceso a Bases de Datos

La Figura 1 muestra la evolución histórica en la forma de acceso a una base de datos, desde los sistemas rígidos en los cuales un nuevo requerimiento supone la modificación del código fuente, pasando por las interfaces gráficas cuya ventaja es la facilidad de uso, los lenguajes de interrogación a BD flexibles, pero orientados a usuarios expertos, hasta llegar a la interrogación en LN que presentaría ambas ventajas: flexibilidad y facilidad de uso para usuarios no expertos.

4.- ORGANIZACIÓN DE LAS PRÁCTICAS

La asignatura tiene asignados 3 créditos teóricos y 1.5 prácticos, lo que supone 15 horas de laboratorio distribuidas en 7 sesiones de 2 horas. Las prácticas se realizan en Prolog (Sicstus Prolog 3.0). Escogemos este lenguaje por dos motivos: 1) es el más usado en el ámbito del PLN, y 2) porque la implementación de un analizador sintáctico descendente es una tarea sencilla utilizando las Gramáticas de Cláusulas Definidas (DCG) que son directamente interpretadas por Prolog. El temario teórico y las sesiones prácticas se planifican de manera que cuando el alumno llega a la sesión ya conoce los conceptos teóricos necesarios para llevarla a cabo. A continuación se detalla el contenido de cada una de las sesiones:

a) Sesión 1: Introducción

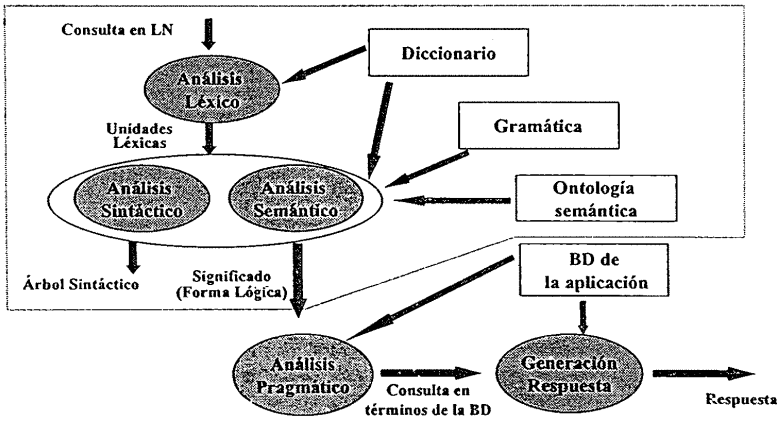


Figura 2: Esquema del sistema de acceso a BD

Durante la primera sesión se presenta el sistema que deberán implementar: una interfaz en LN para la interrogación de una BD, cuyo dominio es información sobre la geografía española. La Figura 2 representa el sistema completo. El alumno implementará los módulos que se encuentran dentro de la zona punteada. Para evaluar su sistema globalmente podrá enlazarlo con el módulo pragmático y de generación de respuesta que se le proporciona.

b) Sesión 2: El Diccionario

Objetivos: 1) conozcer las estructuras básicas para almacenar la información léxica, 2) comprender la importancia de almacenar y recuperar de manera eficiente esta información, 3) implementar estructuras eficientes para el almacenamiento y acceso de esta información ('trie' de palabras).

Tareas: 1) introducir las entradas léxicas para el dominio de la aplicación, 2) implementar el predicado `buscaPalabraTrie(Pal,Lexico)` que devuelve la lista de entradas léxicas (`Lexico`) para una palabra (`Pal`) en un trie implementado mediante un vector. El resto de predicados para acceder al diccionario son proporcionados al alumno.
Cada entrada léxica contiene la estructura de rasgos definida con un termino Prolog, cuyo functor identifica la categoría léxica y cada argumento representa un rasgo. Estos almacenan: información morfológica (género y número), clasificación semántica, interpretación semántica (forma lógica) y restricciones seleccionales. Las entradas léxicas se organizarán en un trie, así por ejemplo la palabra 'caudaloso' se almacenaría en un nodo del trie:

```
nodoTrie([adj(sg,ms,caudaloso(X),[rio]],ramas(A,..,Z,a,..,z,0,..,9))
```

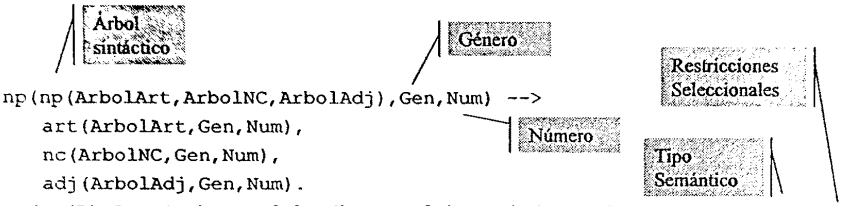
c) Sesión 3: El análisis Léxico - el tokenizador.
Objetivo: construir un segmentador o tokenizador de oraciones en unidades léxicas. Éste será utilizado como primera fase de análisis oracional y accederá al diccionario para verificar la existencia de las unidades léxicas.
Tareas: 1) definir qué es un *token válido*. 2) implementar el tokenizador.
Un token válido será cualquier secuencia de caracteres que se corresponda con: una palabra que comience por mayúsculas o minúsculas, un número o un símbolo de puntuación ('.', '¿', '?').
El resultado del tokenizador será una lista Prolog, cuyos elementos serán los tokens de la oración, por ejemplo,

¿Qué ríos pasan por Valencia? → `[¿,qué,ríos,pasan,por,valencia,?]`

El segmentado se realizará de forma recursiva sobre la lista de caracteres que conforman la oración. Se implementa con la diferencia de listas. Así, el predicado `tokeniza(LPalabras,LChar,LChar1)` debe instanciar en `LPalabras` la lista de tokens obtenida tras analizar la lista de caracteres que forman la oración `LChar`. A partir de la definición de este predicado el alumno desarrolla el resto:

d) Sesión 4 y 5: El Análisis Sintáctico - implementación de la DCG
Objetivo: definir una gramática para un subconjunto del castellano y construir el analizador sintáctico descendente en Prolog.
Tareas: 1) identificar las estructuras sintácticas a partir de un corpus de consultas, 2) construir la gramática utilizando el formalismo de las DCG, 3) incorporar los argumentos necesarios para: generar el árbol sintáctico, la comprobación morfológica, 4) realizar la verificación semántica (asignar un tipo semántico a cada palabra y especificar las restricciones seleccionales).

Los tipos de oraciones son: Interrogativas tipo SÍ/NO (ej. "¿El río Turia pasa por Valencia?"), Interrogativas de sujeto (ej. "¿Qué río pasa por Valencia?") e Imperativas (ej. "Dime los ríos que pasan por Valencia.").
El formalismo que se utilizará para representar la gramática son las Gramáticas de Cláusulas Definidas (DCG).
Se desarrolla completamente una regla simple de sintagma nominal (np) con verificación de concordancias y generación de árbol sintáctico. Tomándola como ejemplo, el alumno define el resto de reglas.




```
np(np(ArbolArt,ArbolNC,ArbolAdj),Gen,Num) -->
    art(ArbolArt,Gen,Num),
    nc(ArbolNC,Gen,Num),
    adj(ArbolAdj,Gen,Num).
nc(nc(P),Gen,Num) --> [P], {buscaPalabraTrie(P, nc(Num, Gen, FL, TS, RS))}.
art(art(P),Gen,Num) --> [P], {buscaPalabraTrie(art(Num, Gen, Tipo, FL))}.
adj(adj(P),Gen,Num) --> [P], {buscaPalabraTrie(adj(Num, Gen, Grado,FL, TS))}.
adj([],_,_).
```

Además de verificar que las oraciones son válidas sintácticamente debe comprobarse su consistencia semántica. Por ejemplo, en la oración "¿ El río Turia pasa por Valencia ?" debe verificarse que, tanto el tipo semántico asociado al sujeto ('El río Turia'), como al objeto ('por Valencia') sean compatibles con las restricciones impuestas por el verbo 'pasar'.
La verificación de la consistencia semántica se realizará por medio del módulo de verificación semántica que se le proporciona al alumno. El predicado verificar(Rasgos, Restricciones) comprueba si la lista de Rasgos de una entidad candidata es compatible con la lista de Restricciones exigidas por otra entidad. Ejemplo:

```
orSiNo --> ['¿'], np(Rasgo), vp(Restriccion),
{verificar(Rasgo,Restriccion)}, ['?'].
```

e) Sesión 6 y 7: El Análisis Semántico - composición de la Forma Lógica
Objetivos: aplicar los conocimientos adquiridos sobre interpretación semántica implementando el mecanismo de composición de la Forma Lógica (FL).
Tareas: 1) incorporar las formas lógicas en las entradas léxicas, 2) añadir los argumentos necesarios en las reglas DCG implementadas en las sesiones anteriores para realizar la composición de la FL.
El significado de cada palabra se expresa mediante una Forma Lógica. Así, un nombre común introduce un predicado unario o binario (para representar complementos habituales). Por ejemplo, la palabra 'río' suele aparecer con un complemento del tipo 'autonomía' tendrá dos significados rio1(X) y

rio2(X,Y), donde X representa un objeto del tipo 'río' e Y un objeto del tipo 'autonomía'. Las entradas léxica para 'río' serían:

```
nc(masc, sing, rio1(X),X:[rio],[]),
nc(masc, sing, rio2(X,Y),X:[rio],[cprep(de):Y:[autonomia]]))
```

donde el último argumento representa las restricciones seleccionales con la forma `tipocomplemento(preposición):Var:Restricc. Semánticas.` Para llevar a cabo el proceso de composición de la FL, las formas lógicas de los constituyentes del lado derecho de una regla gramatical se combinarán para producir la forma lógica del no terminal del lado izquierdo. Para ello se hará uso del mecanismo de Unificación y algunos predicados predefinidos de Prolog. A continuación se ilustra con un ejemplo el proceso de composición de la FL para una gramática que genera oraciones tipo Sí/No.

```
orSiNo(si_no(Sem)) --> ['¿'], np(X,Scope,Sem), vp(X,Scope), ['?'].
np(X,Scope,Sem) --> det(X,Rest,Scope,Sem), nc(X,Rest).
vp(X,Sem) --> v(X,Y,S0), np(Y,S0,Sem).
nc(X,FL) --> [P],{buscaPalabraTrie(P,nc(Num, Gen, FL, X:TS, RS))}.
det(X,Base,Foco,Sem) --> [P],
        {buscaPalabraTrie(P, art(Num, Gen, Tipo, FL)),
        functor(FL,NombrePred,_), Sem =..[NombrePred,X,Base,Foco]}.
v(X,Y,Sem) -->[P],
             {palabra(P, verb(Num,Pers,Tiem,Modo, FL, TS)),
             functor(FL,NombrePred,_), Sem =..[NombrePred,X,Y]}.
```

El atributo `Sem` se instanciará a la FL generada por la expansión del predicado correspondiente. El atributo `Scope` representa el ámbito o foco en la FL.

Los siguientes ejemplos muestran cuál debe ser el resultado de todo el proceso de análisis implementado:

```
¿ Qué ríos de Andalucía desembocan en el mar Mediterráneo ?
preg(Y,rio2(Y,andalucía)&existe(X,mar1(X)&X=mediterráneo,desembocar(Y,X)))

¿ Qué ríos desembocan en Andalucía ?
preg(X,rio1(X)&desembocar(X,andalucía))
```

5.- BIBLIOGRAFÍA

[Allen95] J. Allen. Natural Language Understanding. 2nd.ed. Benjaming Cummings series in Computer Science. 1995

[Moreno98] L. Moreno, A. Molina. La disciplina del Lenguaje Natural en el título de Ingeniero en Informática. JENUI'98. Enginyeria i Arquitectura La Salle, 1998.
[Covington94] M.A. Covington. Natural Language Processing for Prolog Programmers. Prentice Hall. 1994
[Matthews98] C. Matthews. An introduction to Natural Language Processing Through Prolog. Longman, 1998.