# TRATAMIENTO LEXICOGRAFICO DE UN SISTEMA DE TRADUCCION AUTOMATICA

*Isabel Zapata Domínguez*
*Enrique Torrejón Díaz*

Centro de Investigación UAM-IBM
Santa Hortensia 26 - 28
28002 Madrid
isabel@emdcci1 1 .bitnet
enrique@emdcci1 1 .bitnet

**Resumen**

La comunicación que se presenta describe los aspectos mas relevantes del tratamiento lexicográfico de un prototipo multilingüe de Traducción Automática basado en un modelo de transferencia controlado por la información léxica. En primer lugar, se explica con cierto detalle el formalismo que se emplea en los diccionarios del sistema. A continuación, se exponen las herramientas desarrolladas para solucionar el problema de la extracción automática masiva de información lingüística y su transformación al formalismo descrito.

## Introducción

En la última década, la información contenida en los diccionarios electrónicos (MRD)[1] ha sido objeto de un progresivo interés por parte de los investigadores de PLN[2] debido a la creciente necesidad de incorporar datos lingüísticos, de una forma masiva y sistemática, en las bases de conocimiento de los sistemas de procesamiento del lenguaje. Esta necesidad es especialmente acuciante cuando se trata de desarrollar un sistema de Traducción Automática (TA) basado en un modelo de transferencia controlado por la información léxica. Como consecuencia de esto, la Lexicografía Computacional se ha destacado como disciplina dentro del ámbito de la Lingüística Computacional.

Siguiendo esta tendencia, IBM ha realizado un gran esfuerzo en la investigación de tecnología para sistemas de PLN y su implementación. Una de las iniciativas a la que se presta especial atención es el desarrollo de un *shell* de Traducción Automática, **LMT (Logic-programming based Machine Translation)[3] [McCord 89a, 89b, 89c, 89d],** cuyo autor es M. C. McCord del T. J. Watson Research Center. La tarea de crear prototipos de TA para varios pares de lenguas se ha asignado a diferentes equipos de otros tantos Centros de Investigación en Europa y EEUU. El grupo del Centro de Investigación de Madrid trabaja, en concreto, en los pares de lenguas inglés-español y español-inglés.

---

[1] *Machine Readable Dictionaries.*

[2] *Procesamiento del Lenguaje Natural.*

[3] *Logic-programming based* se refiere a que está codificado en lenguaje Prolog.

LMT es un sistema modular, basado en la arquitectura *transfer* que consta de tres módulos principales: uno de análisis para la lengua fuente (ver *Slot* Grammar **[McCord 90]**), uno de transferencia de la lengua fuente a la destino y otro para la generación de la lengua destino. El módulo transfer resuelve dos tipos de transferencia: composicional (también llamada *léxica) y estructural.*

Los diccionarios para LMT son bases de datos independientes de los módulos antes mencionados. En estas bases de datos, las entradas están declaradas en un formalismo que se denomina *LMT LEF*[4] y que se describirá con más detalle en la sección "LMT LEF: la formalización de las entradas léxicas" en la página 3. Cada entrada contiene datos para la fase de análisis y para la fase de *transfer*. Toda esta información en LMT LEf es la que el *parser* accede en tiempo de ejecución cuando construye las estructuras asociadas a cada una de las palabras que constituyen la oración analizada. Estas estructuras serán utilizadas:

- por la gramática para construir los arboles de análisis aplicando reglas sintáctico-semánticas, y
- por el módulo de *transfer* para proceder a las transformaciones léxicas y estructurales que sean necesarias.

Desde el punto de vista organizativo, los diccionarios se han clasificado en **terminológicos y de propósito general,** cada uno de los cuales es una base de datos distinta. Con esta clasificación se pretende poder adaptar el sistema de traducción a cualquier tipo de texto de una forma eficaz y versátil. Hasta el momento, se está elaborando para el prototipo un diccionario con terminología utilizada en manuales técnicos de IBM y un diccionario general que contiene el vocabulario básico.

Los problemas que plantea el diseño de un prototipo de TA con relación a la información lexicográfica son varios: a) de dónde extraer dicha información, b) cómo obtener un gran número de entradas una vez elegidas las fuentes y c) cómo transformar esa información al formalismo elegido (en este caso, LMT LEF).

En cuanto a la búsqueda de las fuentes, dos son las tendencias que actualmente se muestran más relevantes:

- la **reutilización** de la información contenida en los diccionarios tanto impresos como electrónicos. Normalmente, la versión en soporte magnético contiene exactamente la misma información que se encuentra en la versión impresa, además de una serie de tags *de edición* que sirven para dar forma al diccionario impreso. En cualquier caso, los diccionarios presentados de esta manera tienen el inconveniente de estar orientados para que los utilice un usuario humano, que es capaz de entender e inferir información que no está explícitamente declarada. Los diccionarios no están preparados para que un sistema automático pueda utilizar su contenido de una forma directa.

- la extracción de información lexicográfica a partir de **corpus textuales,** utilizando métodos sintácticos **[Anick/Pustejovsky 90]** o **estadísticos [Calzolari/Bindi 90].** Este planteamiento es más reciente y trata de cubrir las deficiencias planteadas con el método anterior, ya que llega un momento en que los diccionarios se quedan obsoletos, mientras que el procesamiento continuo de corpus de texto real refleja los usos más recientes de la lengua. Para este procesamiento masivo, hoy en día se dispone de herramientas informáticas capaces de manejar grandes cantidades de texto, por lo que el problema computacional está resuelto.

La solución adoptada en el prototipo inglés-español es la reutilización de diccionarios electrónicos, y viene determinada porque

- se tiene acceso de manera inmediata a MRDs monolingües y bilingües,

[4] Lexical Entry Format

- no se dispone de corpus de textos monolingües en general y bilingües en particular, adecuados para la extracción de esta información, ni de los programas estadísticos que pudieran tratarlos,
- la línea de trabajo del *Lexical Research Group* del T.J. Watson Research Center aconseja la explotación de estas fuentes. Con respecto a los corpus textuales, todavía es necesaria mucha investigación para la implementación de programas totalmente fiables que transfieran la información estadística a un formalismo concreto.

Las fuentes con las que trabajamos son las versiones electrónicas del *Collins English-Spanish Dictionary* **[Collins 79]**, del *Longman Dictionary of Contemporary English (LDOCE)* **[Longman 81]** y del *Webster Seventh New Collegiate Dictionary (Webster7)* **[Webster 87]** . Además para el proceso de validación de las entradas se consultan otros diccionarios impresos como el *Collins Cobuild English Language Dictionary* y el *Larousse English-Spanish Dictionary*.

Con respecto a como obtener un gran número de entradas una vez elegidas las fuentes, contamos con herramientas que transforman la información originalmente en soporte magnético en **Bases de Datos Léxicas (BDL).**

Por último, con relación a cómo transformar los registros de una BDL en formato LMT LEF, se han implementado programas que llevan a cabo esta tarea.

En la sección "Extracción automática de información léxica" en la página 9 se describen con más detalle los procedimientos mencionados en los dos párrafos anteriores.

## LMT LEF: la formalización de las entradas léxicas

En la introducción se ha mencionado que LMT es un sistema de traducción controlado por el léxico. Esto significa que,

- el léxico proporciona datos a la gramática, especificando posibles partes de la oración (que se expresan como *functores)* para cada entrada y los argumentos que expresan el papel lógicosintáctico *(slots)* de cada parte de la oración con respecto a otros elementos
- se pueden especificar las traducciones de cada uno de los sentidos de la entrada según criterios contextuales, semánticos, sintácticos y morfológicos, y
- el formalismo tambien permite especificar determinadas transformaciones estructurales mediante un operador *de transformación léxica (lxt)* que cambia los *slots* de la lengua fuente por otros distintos de la lengua destino.

La estructura de una entrada en un diccionario LMT es, de forma simplificada, la siguiente:

```
palabra_índice < análisis1 < transfer1
               < análisis2 < transfer2
                 ......
```

Más específicamente, el formalismo de las entradas tiene un aspecto semejante a una cláusula de Prolog:

```
palabra_indice < functor1(Argumentos) < t(Condiciones ? Traducción; Casos)
               < functor2(Argumentos) < t(Condiciones ? Traducción; Casos)
                 ......
```

donde:

- **palabra_índice** es la forma base de las palabras que, en nuestro prototipo coincide con las entradas de un diccionario en el sentido tradicional. Esta forma base es la que se obtiene como resultado del análisis morfológico
- **functor** es la abreviación de los nombres de las partes de la oración (n para nombres, v para verbos, etc.)
- **Argumentos** es información **para cada una** de las partes de la oración de la lengua origen acerca de: a) los *sentidos* o acepciones; *b) los rasgos semánticos*, y c) los *slots*, que se refieren a información lógico-sintáctica.
- **Condiciones** son operadores o combinaciones booleanas de operadores que comprueban la existencia de determinados argumentos en la estructura de análisis de la oración fuente para discriminar entre diferentes traducciones
- **Traducción** es la traducción de la *palabra_índice*
- **Casos** se refiere al campo donde se almacena la información sobre las relaciones lógico-sintácticas de la lengua destino.

Para aclarar el significado de estas definiciones, nos centraremos en la formalización de algunos ejemplos de nombres y verbos[5] **[McCord/Schwall 90]**.

### Formalización para verbos

La codificación de verbos se basa en la formalización de **[Quirk et al. 85]** que retoma la tradición iniciada por **[Tesnière 59], [Gruber 65] y [Jackendoff 72]**. Siguiendo su clasificación, se distinguen los siguientes tipos de verbos segun su uso: intransitivos, transitivos y ditransitivos. Cualquier verbo en uno de estos tres usos puede subclasificarse a su vez en frasal preposicional[6] , frasal adverbial, frasal preposicional-adverbial u otros.

### Verbos intransitivos

Si un verbo con uso intransitivo tiene tambien un uso transitivo y se traduce, en ambos casos, de la misma forma, no se codificaran independientemente. Por ejemplo,

---

[5] Debido a la extensión y detalle con los que los autores han querido explicar la formalización para nombres y verbos, en esta comunicación no se tratarán las demás partes de la oración.

[6] El término *frasal* ha sido traducido del inglés *phrasal*. Atendiendo al criterio de **[Quirk et al. 85]**, se entiende que un verbo frasal es una combinación de verbo simple y monosilábico más una partícula (adverbio, preposición, adjetivo) con significado no composicional. Por ejemplo, el verbo *put* con el adverbio *off* constituye el verbo frasal *put off* que significa *posponer o desanimar*, según el contexto. En muchos aspectos, los verbos frasales funcionan como una única palabra aunque, bajo ciertas condiciones, se pueden insertar otros elementos (objetos directos, adverbios) entre el verbo y la partícula.

```
destroy < v(obj) < t(destruir).

The enemy destroy
El enemigo destruye

The enemy destroy the city
El enemigo destruye la ciudad
```

Por el contrario, véase el ejemplo de *act* para el que se codifican separadamente ambos usos porque tienen distintas traducciones:

```
act < v       < t(actuar)
    < v(obj1) < t(representar).

I can't take her seriously because she always seems to be acting
Yo no puedo tomarla en serio porque ella siempre parece estar actuando

Olivier is acting Othello tonight
Olivier está representando Otello esta noche
```

En este caso es interesante notar que al *slot* que marca el uso transitivo, **obj,** se le añade el digito **1** para indicar *obligatoriedad* de la existencia de un objeto directo. Sólo cuando este objeto existe, **act** sera traducido por *representar*.

En los ejemplos que siguen, los verbos intransitivos llevan marcados sus complementos más frecuentes:

```
come < v(comp(ing|inf)) < t(so(inf) ? venir;infa | venir).

trouble < v(compl(inf)) < t(se::molestar;infen)
        < v(obj) < t(molestar).

The girl came running to her mother
La niña vino corriendo a su madre

Don't trouble to write when I am gone
No se moleste en escribir cuando yo me haya ido
```

En la entrada **come,** el *slot* **comp(ing)** indica que la construcción de gerundio es el complemento habitual. Por último, en **trouble** el complemento de infinitivo **(compl(inf))** siempre se traduce como una construcción de infinitivo introducida por la preposición *en* **(infen).**

Los verbos intransitivos en los que se quiere especificar algún complemento preposicional habitual, se declaran marcando el *slot* **p(xxx)** con la(s) preposición(es) pertinente(s).

```
worry < v(p(for|about)) < t(se::preocupar;pc(por))
      < v(obj1(n|inf).iobj) < t(preocupar).
```

**Worrying about your health** can make you ill
**Preocuparse por su salud** puede ponerle enfermo

En la codificación de los verbos intransitivos frasales, las partículas (adverbios) se especifican en la parte de análisis dentro de un *slot* **pt(xxx)**.

```
go < v(comp(p(to))) < t(ir;pc(a))
   < v(pt1(about)) < t(virar)
   < v(pt1(ahead|along).p(with)) < t(pt(ahead).p(with) ? empezar;pc(con) | pt(ahead).x ? seguir |
       pt(along).p(with) ? estar:::'de acuerdo';pc(con) | pt(along).x ? continuar)
   < v(pt1(back).p(to)) < t(pt(back).st(tm) ? se::remontar;pc(a) |
                                  p(to) ? volver;pc(a) | regresar)
   < v(obj1(p(into))) < t(entrar;pc(en))
   < v(obj1(ing).pt1(on)) < t(continuar).
```

En esta entrada, se distinguen cuatro verbos frasales intransitivos: **go about** *(virar)*, **go ahead** (*empezar con*, si le sigue un complemento preposicional con *with, o seguir* en caso contrario), **go along** *(estar de acuerdo con*, si le sigue un complemento preposicional con *with, o continuar* en caso contrario) y **go back** *(remontarse* a, si le sigue un complemento preposicional con *to* y el núcleo del complemento tiene la marca semántica **tm** por **time**, o *volver* a, si le sigue un complemento preposicional con *to, o regresar* en último caso). Además, este ejemplo muestra otras traducciones para **go**, aunque la codificación no está totalmente completa.

## Verbos transitivos

En español, los objetos directos de persona son sintagmas preposicionales introducidos por *a*. Por esta razón, aquellos verbos que puedan construirse de este modo deben verificar si el objeto directo tiene rasgo h, en cuyo caso se incluirá la preposición *a* con **pc(a)**.

```
see < v(obj(n|fin)) < t(h ? ver;pc(a) | ver).

I saw the woman in the house
Yo vi a la mujer en la casa

Let me see your ticket, please
Déjeme ver su billete, por favor
```

Para codificar verbos transitivos con objeto directo cuyo núcleo es un infinitivo se marca el *slot* **obj(inf)** y se especifica **infde**, **infa** o nada en el campo de *casos*, dependiendo de si el objeto directo se traduce *por de* máás infinitivo, *a* más infinitivo o simplemente infinitivo.

```
omit < v(obj(n|inf|ing)) < t(so(n) ? omitir | dejar;infde).

begin < v(pl(with))       < t(empezar;pc(con))
      < v(obj(n|inf|ing)) < t(empezar;infa)
      < v(obj1.pl(by))    < t(empezar;pc(por)).

deserve < v(obj(inf)) < t(merecer).

Please, don't omit to lock the door
Por favor, no deje de cerrar la puerta
```

They **began to dance**
Ellos **empezaron a bailar**

She **deserved to win** because she was the best
Ella **mereció ganar** porque ella era la mejor

La traducción del *slot* **obj(ing)** puede variar. Cuando el verbo en español se traduce por un infinitivo compuesto, en el campo de *casos* se especifica **infpast.**

```
forget < v(obj(p(about)|n|fin|inf|ing)) < t(p(about) ? se::olvidar;pc(de) |
                                            f(verb(prespart)) ? olvidar;infpast | olvidar).
```

I'll never **forget finding that old coin in my garden**
Yo nunca **olvidaré haber encontrado esa moneda antigua en mi jardín**

Otra posibilidad es traducir el gerundio por un infinitivo con la preposición *de*, con lo que hemos de introducir la marca **infde.**

```
finish < v(obj(n|ing)) < t(terminar;infde).
```

I haven't **finished reading the book** yet
Yo no he **terminado de leer el libro** todavía

Cuando se traduce a un infinitivo con la preposición a, se especifica **infa.**

```
start < v(pl(with|from)) < t(comenzar;pc(con|desde))
      < v(obj.comp(ft)) < t(copy.x ? iniciar | solution.x ? iniciar | arrancar)
      < v(objl(inf|ing)) < t(empezar;infa)
      < v(objl.ptl(up)) < t(arrancar).
```

**It has started raining**
**Ha empezado a llover**

A veces, es necesario especificar varios complementos preposicionales para un mismo verbo transitivo, que se traduce por distintos verbos con cada preposición.

```
slide < v < t(se::deslizar)
      < v(objl.p(in|into|out.of|toward))
      < t(p(in) ? deslizar;pc(en)  | p(into) ? introducir;pc(en)    |
          p(out.of) ? sacar;pc(de) | p(toward) ? deslizar;pc(hacia) | deslizar)
      < v(objl.ptl(in).comp(n)) < t(introducir).
```

Por último, es necesario introducir el modo en que se declara el objeto directo doble, es decir, aquel complemento cuya primera parte es una expresión nominal y la segunda un infinitivo (con o sin *to, I helped him (to) clean the windows),* una forma de gerundio *(I saw the man leaving)* o un participio pasado (I *have the house built).* La formalización de objeto directo doble para los verbos del ejemplo, sería:

```
help < v(obj1.compl(binf|inf)) < t(ayudar;dat.infa).

see < v(obj1.compl(en|ing|binf))
    < t(lxt(obj(np).comp(ing),obj(np).comp(inf)) ? ver | ver).

have < v(obj1.compl(en))
     < t(lxt(obj(np):X.comp(en),empty.obj(inf):addright(obj(*):X)) ? hacer | tener).
```

Para el verbo **help,** la trasformación indica que el objeto directo pasa a ser un objeto indirecto (caso dativo) y el complemento de infinitivo pasa a ser una construcción de infinitivo introducida por la preposición a ((Yo) le ayude ***a limpiar*** las ventanas).

Para el verbo **see**, la transformación indica que, cuando el objeto directo es un sintagma nominal complementado a su vez por una forma de gerundio, este último pasa a ser un infinitivo *((Yo) vi al hombre salir)*. Es importante destacar la función del operador **lxt**[7] que hasta el momento no había sido mencionado. Este operador permite especificar transformaciones estructurales activadas por la misma entrada léxica, de modo que estas transformaciones no tienen que ser programadas específicamente mediante reglas en el módulo de *transfer*.

En la entrada **see**, el operador **lxt** toma como primer argumento la lista de *slots* **obj(np).comp(ing)** y la transforma en la lista de *slots* **obj(np).comp(inl)** para la lengua destino. Se puede observar la correspondencia entre los *slots* en ambas listas, de manera que el primer *slot* de la primera lista se transforma en el primer *slot* de la segunda y lo mismo para los segundos *slots*.

Para el verbo **have,** la transformación indica que, cuando el objeto directo es un sintagma nominal complementado a su vez por una forma de participio pasado, este último se transforma en un infinitivo en función de objeto directo del verbo **have,** y el sintagma nominal pasa a ser el objeto directo del infinitivo *((Yo) hago construir la casa)*. La variable **X** en el primer argumento de **lxt** pone una marca de referencia al *slot* **obj(np)** que se transforma en un *slot* vacio (**empty**). El *slot* **comp(en)** se transforma en **obj(inf)** y a este último se le añade a la derecha (**addright**) un *slot* de tipo **obj(*)** que se rellena con la información del sintagma nominal referenciado por **X**.

### Verbos ditransitivos

Para los verbos ditranstitivos, es válido todo lo dicho anteriormente respecto a la especificación de los *slots* y del campo de *casos*. Sin embargo, ahora debe tenerse en cuenta que es obligatorio especificar una estructura de *slots* que comprenda tanto el objeto directo como el indirecto. Por ejemplo, **obj.iobj, obj(inf).iobj, obj(fin)..obj, obj(fin|inf).iobj, obj(p(xxx)).iobj,** y otras posibles combinaciones.

```
give < v(obj.iobj)      < t(dar).
tell < v(obj(fin).iobj) < t(decir).
```

---

[7] lxt, forma abreviada de *lexical transformation*. El formalismo del operador lxt es

```
lxt(slots_origen,slots_destino)
```

y se especifica en la parte de *condiciones*

```
t(Condiciones ? ...).
```

En general, esta transformación cambia los *slots* de la lengua origen a cualquier estructura de *slots* que se quiera de la lengua destino. El tipo de transformaciones que permite especificar son: desaparición de *slots*, adición de *slots*, cambio de *slots*, etc.

**Give me the baby**
**Dame el bebe**

He **told me his name**
**El me dijo su nombre**

He **told me his name was John**
**El me dijo que su nombre era John**

En todos los usos de los verbos, existe la posibilidad de utilizar los *sentidos y rasgos semánticos* para desambiguar entre acepciones u homógrafos de los mismos.

```
run < v < t(h ? correr | cmpt ? se::ejecutar | funcionar)
    < v(obj1) < t((computer|system) ? poner:::'en marcha' | cmpt ? ejecutar | procesar)
    < v(obj1(p(on))) < t(funcionar;pc(con)).
```

**Run**, como verbo intransitivo, se traduce por *correr* cuando el sujeto es **h**; por ejecutarse si el sujeto lleva la marca **cmpt** *(computing)* y por *funcionar* en cualquier otro caso. Como verbo transitivo, se traduce por *poner en marcha* cuando el objeto directo tiene como núcleo las palabras *computer o system; ejecutar* cuando el objeto directo está marcado como **cmpt,** o *procesar* en los demás casos.

## Formalización para nombres

- La forma más sencilla de declarar un nombre aparece en el ejemplo siguiente, que sólo consta de entrada, parte de la oración y traducción:

```
sun < n < t(sol).
```

- El tratamiento simultáneo de la información en los *slots* y en el campo de *casos* para la lengua origen y destino, respectivamente, se puede ver cuando un nombre es complementado por un sintagma preposicional introducido por **of** que normalmente se traducirá por *de*. La opción de que **circulation** tenga un complemento preposicional con **of** se indica en el formalismo con el *slot* **obj.**

```
circulation < n(obj) < t(circulación).
```

This magazine has a **circulation of** 400.000 mainly in the north of England
Esta revista tiene una **circulación de** 400.000 principalmente en el norte de Inglaterra

En caso de que el nombre tenga más de un complemento preposicional introducido por diferentes preposiciones, se especifican estas (**p(xxx|yyy)**) y sus correspondientes traducciones en el campo de *casos* (**pc(www|zzz)**).

```
disagreement < n(p(between|with)) < t(desacuerdo;pc(entre|con)).
```

I am in total **disagreement with** you
Yo estoy en total **desacuerdo con** usted

Para codificar sintagmas preposicionales cuyo núcleo es un infinitivo se marca el *slot* obj(inf) y se especifica **infde, infa, infen, infor o infpara** en el campo de *casos*, dependiendo de si la preposición se traduce por de, *a, en, por o para*, respectivamente.

```
reason  < n(obj(fin|inf)) < t(razón;(pc(de)|infpara)).
attempt < n(obj(inf|n))   < t(intento;(infde|pc(de))).
```

**There is some reason to believe he will come**
**Hay alguna razón para creer que él vendrá**

**An attempt to climb the mountain**
**Un intento de escalar la montaña**

Si lo que complementa al nombre es una oración completiva (**obj(fin)**), se marcará la correspondiente preposición, normalmente *de*, como en el ejemplo siguiente:

```
desire < n(obj(fin|inf)) < t(deseo;(pc(de)|infde)).
```

**A desire that she should go**
**Un deseo de que ella debería ir**

Si lo que se quiere especificar es *traducción vacía* para la preposición destino, se marcara con **x** en el campo de *casos*. El *slot* **obj(ft)** se utiliza para especificar una oración subordinada completiva introducida por **for**.

```
problem < n(obj(p(in|with)|fin|inf|ft)) < t(problema;(pc(en|con)|pc(de)|infde|x)).
```

**The problem for the system to run the program is hard**
**El problema de que el sistema ejecute el programa es difícil**

Si el nombre tiene dos o más posibles complementos preposicionales, cada uno de ellos debe ser marcado explícitamente con un *slot*.

```
conversion < n(obj.p(from).p(to)) < t(conversión;pc(de).pc(a)).
```

**Conversion of your heating system from coal to gas will be costly**
**La conversión de su sistema de calefacción de carbón a gas será costosa**

- Los *rasgos semánticos* de un nombre se pueden tener en cuenta a la hora de formalizar otras partes de la oración, especialmente verbos y adjetivos, ya que puede determinarse el significado de estos según los rasgos semánticos del nombre al que acompañen. Por ejemplo, si tenemos un nombre marcado con el *rasgo semántico* **h** (o **humano**).

```
programmer < n(h,nil) < t(programador).
```

podría marcarse que el adjetivo **old** se tradujera de forma distinta cuando modifica a un nombre con o sin rasgo **h**. Lo mismo sucedería con el verbo **question,** donde se preven dos traducciones distintas dependiendo de si el objeto directo lleva dicho rasgo semántico.

```
old < adj < t(h ? viejo | antiguo).

question < n(p(about))    < t(pregunta;pc(sobre))
         < v(obj(n|fin)) < t(h ? interrogar;pc(a) | poner:::'en duda').
```

The **old** programmer is here / El programador viejo está aquí
The **old** program is here / El programa antiguo está aquí

The system **questions** the programmer / El sistema interroga al programador
The researcher **questions** his honesty / El investigador pone en duda su honestidad

- Los diferentes *sentidos* de la palabra **way** deben distinguirse cuando éstos se vayan a utilizar para desambiguar la traducción de un verbo o adjetivo. Esta distinción se hace en el formalismo añadiendo un dígito a cada acepción. Por ejemplo, **way**, cuando está modificado por una oración completiva sin *that* (**finv**) o por una construcción de infinitivo (**inf**), se traducirá por *forma* (acepcion **way2**), y por *camino* (acepción **way1**) en cualquier otro caso.

```
way < n(way1,loc,p(to|of)) < t(camino;pc(a|de))
    < n(way2,loc,obj1(finv|inf)) < t(forma;(pc(en)|infde)).

lead < v(obj) < t(way1 ? mostrar | encabezar).
```

El verbo **lead** se traduce como *mostrar* en la construccion *lead the way (mostrar el camino)*, donde **way** se refiere a **way1**. En cualquier otro caso se traduce por *encabezar*.

Los ejemplos mostrados reflejan únicamente un aspecto parcial de las posibilidades que LMT LEF permite en la especificación de los nombres. Es también muy interesante la versatilidad de este formalismo en la declaración de otras construcciones como las *lexicalizaciones nominales*, los nombres propios y geográficos, los acrónimos, etc.

## Extracción automática de información léxica

En la sección anterior se ha descrito brevemente LMT LEF para nombres y verbos, y tambien se ha apuntado que, con un formalismo muy similar, se declaran las restantes partes de la oración (evidentemente, los nombres y los verbos soportan la mayor carga sintáctico-semántica en la oración). Como es obvio, no es fácil especificar una entrada completa si no se cuenta con una fuente de donde se puedan extraer todos los usos de dicha palabra, es decir, un diccionario o un corpus. También es evidente que la codificación manual de todas las entradas a partir de uno o varios diccionarios impresos llegaría a ser una tarea ímproba que requiriría el esfuerzo de muchos lexicógrafos. Además, la consistencia del formalismo no estaría de ningún modo garantizada, ni aun en el mejor de los casos en el que el trabajo lo llevara a cabo sólo un lexicógrafo experto. Por otro lado, con cualquier cambio en el formalismo (hemos de pensar siempre que LMT es un sistema que, actualmente, se mejora día a día) o, incluso, con las modificaciones normales que la lengua experimenta con el tiempo, el diccionario requeriría una labor de revisión que, a priori, se presenta muy difícil.

En nuestro proyecto se ha prestado especial atencion al desarrollo de herramientas que a) reduzcan el coste de construcción de diccionarios, b) incrementen la fiabilidad, c) permitan el

uso de datos lexicos en diferentes sistemas de PLN y d) conecten los diccionarios con nuestro sistema de traduccion.

Para cumplir todos estos objetivos se ha diseñado, en primer lugar, un *shell* denominado **Dictionary Entry Parser (DEP) [Neff/Boguraev 90]**, escrito en Prolog, que permite analizar cualquier MRD y obtener la información de las entradas del diccionario, estructurada según una *plantilla*[8] y almacenada en una base de datos. DEP utiliza como entrada una cadena de caracteres tipográficos sin estructura, es decir, un diccionario electrónico y, mediante la consulta de una *gramática específicamente* diseñada para el diccionario en cuestión, produce como salida representaciones estructurales explícitas para cada entrada individual, que pueden almacenarse en un registro como árboles o en una BDL. En esta línea, los autores han diseñado una gramática que analiza el *Collins English-Spanish Dictionary*, de manera que se tiene este diccionario almacenado en una BDL, a la que cualquier sistema puede acceder para su consulta. Tomemos como ejemplo la entrada **bat** y sigamos todos los pasos del proceso de construccion de la BDL. En la figura se muestra la entrada tal y como aparece en el diccionario impreso.

**bat**[1] [bæt] n (*Zool*) murciélago m; **to be —s, to have —s in the belfry** estar chiflado.
**bat**[2] [bæt] **1** n (*eg cricket* —) maza *f*, palo m; (*Baseball*) bate m; (*fam*) golpe m; **off one's own** — sin ayuda de nadie; **right off the** — de repente, sin deliberación.
**2** *vt* (*fam*) golpear.
**3** *vi* (*Baseball*) batear.
**bat**[3] [bæt] *vt*: **without —ting an eyelid** sin pestañear.

Forma impresa de la entrada **bat**

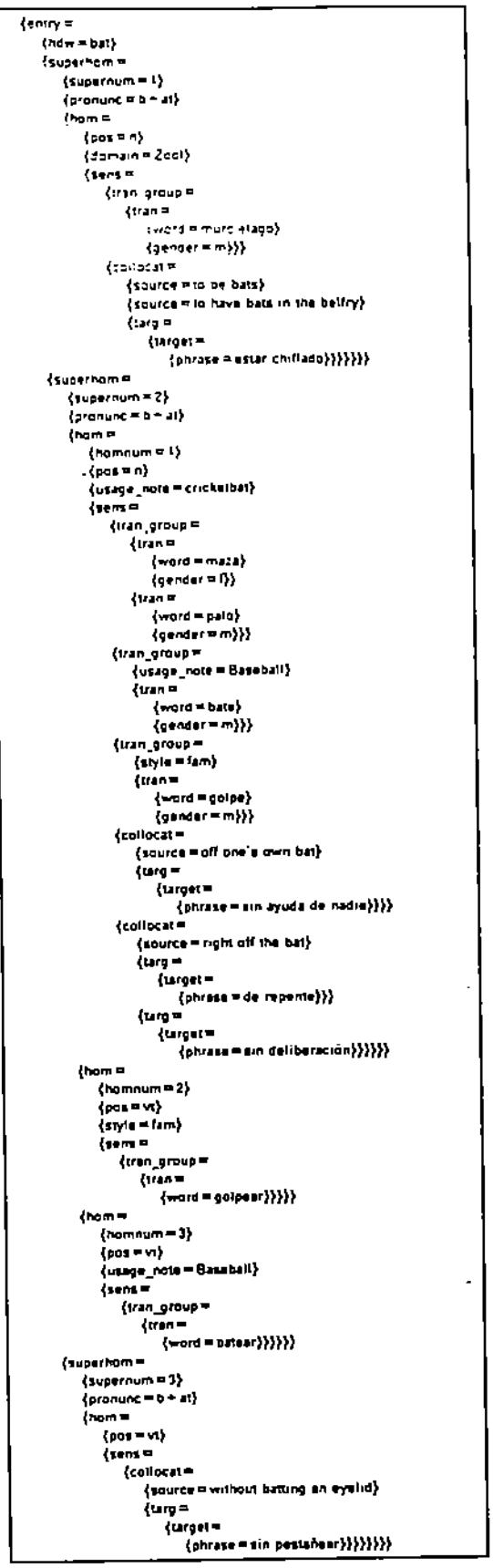
Esta entrada se compone de tres homógrafos identificados por los superíndices 1, 2 y 3 *(supernum* en la plantilla). En el homógrafo **bat**1 se distingue el campo donde se transcribe la pronunciación fonética *(pronunc);* una etiqueta para la parte de la oración *(pos);* una etiqueta de área temática *(domain);* la traducción *(word);* el género de la traducción *(gender);* una o más frases donde *se* ejemplifican usos especiales de la palabra traducida o *collocations (collocat).* El homógrafo **bat**2 tiene tres funciones gramaticales *(homnum),* cada una de las cuales tiene, a su vez, una etiqueta de parte de la oración *(pos),* notas de uso *(Baseball),* de estilo *(fam)* y traducciones y frases típicas. El homógrafo **bat**3 presenta la particularidad de que se usa como verbo transitivo en el contexto de una *frase hecha* y se proporciona únicamente la traducción de la misma.

La figura muestra la información de la entrada **bat** tal y como se almacena en soporte magnético. Se puede apreciar la existencia de signos especiales que controlan cambios tipográficos (de tipo de letra, caracteres especiales, e incluso caracteres no imprimibles).

%1Ñ5ffiÑ2b+atÑ5" Ñ6nÑ5 (Ñ6ZoolÑ5) murciélago Ñ6mÑ5: Ñ1to be $s, to have $s in the belfry Ñ5estar chiflado. %2Ñ5ffiÑ2b+atÑ5" Ñ31Ñ6 n Ñ5(Ñ6eg cricket Ñ5$) maza Ñ6fÑ5, palo Ñ6mÑ5; (Ñ6BaseballÑ5) bate Ñ6mÑ5; (Ñ6famÑ5) golpe Ñ6mÑ5; Ñ1off one's own Ñ5 $ sin ayuda de nadie; Ñ1right off the Ñ5$ de repente, sin deliberación. %%Ñ 32Ñ6 vt Ñ5(Ñ6famÑ5) golpear. %%Ñ33Ñ6 vi Ñ5(Ñ6BaseballÑ5) batear. %3Ñ5ffiÑ2b+at Ñ5" Ñ6vtÑ5: Ñ1without $ting an eyelidÑ5 sin pestañear.

Entrada **bat** en MRD

[8] En inglés, *template.*




Plantilla de la entrada bat en una BDL

Una vez que DEP procesa esta cadena de caracteres no estructurados en soporte magnético con la gramática diseñada para analizar el *Collins English-Spanish Dictionary*, obtiene una *plantilla* que es un homógrafo de la estructura de la entrada según la interpretación que hace la gramática de la misma. Esta plantilla para la entrada **bat** se puede ver en la figura siguiente.

DEP proporciona un formalismo de tipo DCG[9] de definicion de gramáticas. Dado que no se procederá a explicar con detalle este formalismo, baste como ejemplo la declaración de una regla de la gramática para el *Collins English-Spanish Dictionary*, que define los subcampos que pueden aparecer debajo de un homógrafo en el árbol: pronunciación; referencias cruzadas; comentarios referidos a notas de uso, de estilo, geográficas y de área temática; forma negada, plural o gerundio de otra palabra a la que se hace referencia; forma de pretérito o participio pasado de otra palabra a la que se hace referencia; funciones gramaticales con sus subcampos respectivos (bajo *homlist)*.

```
superhom ==> opt(pronunc) :
             opt(comment_rec : xrefform) :
             opt(xrefform) :
             opt(negform) :
             opt(plform) :
             opt(gerform) :
             opt(pretptform) :
             opt(-cl) :
             opt(homlist) :
             opt(unparsable(superhom)).
```

Regla de la gramática para el *Collins English-Spanish* Dictionary

La aplicación de la **regla superhom** fuerza la ejecución de otras reglas tales como **pronunc, xrefform,** etc. El hecho de que estas reglas sean llamadas como argumento del operador opt indica que no es obligatorio que se encuentren estos campos en la entrada, es decir, la plantilla general, que refleja la estructura de cualquier entrada del *Collins English-Spanish* tiene muchos más campos y etiquetas que la que muestra la plantilla del ejemplo. Normalmente las entradas de un diccionario contienen información de muy diversa índole y de ahi que las plantillas de las entradas completen distintos campos.

[9] *Definite Clause Grammar.*

Resta ahora describir cómo acceder a los datos en la BDL del *Collins English-Spanish* y cómo transformarlos a LMT LEF. Para ello se ha implementado un módulo que proporciona el acceso a la BDL en tiempo real, convirtiendo los datos almacenados en dicha BDL al formalismo para el que se diseñe (en nuestro proyecto, LMT LEF).

Este módulo se denomina COLLES, COLLGE o COLLEG cuando accede al Collins inglés-español, alemán-inglés o inglés-alemán, respectivamente. Cada módulo consta de un componente, COLLXY, que es común para todos, y un componente particular para cada par de lenguas, donde se declaran las reglas que transforman la BDL a LMT LEF. LMT puede llamar al módulo COLLES en tiempo de ejecución, pasándole como argumento la forma base de la palabra de la que quiere obtener su declaración LMT LEF. Despues esta declaracion es interpretada por la gramatica y el módulo de *transfer* de LMT tal y como se ha explicado en la seccion anterior.

Otros grupos de trabajo en IBM han realizado gramáticas para otros diccionarios electrónicos como el LDOCE **[Boguraev/Briscoe 89]**, Webster7 **[Klavans et al. 91a, 91b]**, *Collins German-English EnglishGerman* **[Neff/McCord 90]**, y los autores están desarrollando una gramática para el *Collins Spanish-English Dictionary*.

Además de la LDB del *Collins English-Spanish,* sólo utilizamos la del LDOCE para la codificación de entradas, ya que este diccionario~posee un repertorio de códigos para subcategorizar sintácticamente todas las partes de la oración, que, en el caso de los verbos, es especialmente detallada y útil para nuestro propósito. Actualmente, una parte del proyecto consiste en extraer los códigos gramaticales de la base de datos léxica del LDOCE **[BoguraevlBriscoe** 87, 89] y transformar esta información a formato LMT LEF para obtener la subcategorización de los verbos. Por supuesto, esta formalización no proporciona las traducciones de los mismos, por lo cual habra que consultar otras fuentes y contrastar los resultados con la codificación obtenida del *Collins English-Spanish.*

Como ejemplo ilustrativo de este proceso[10] , consideremos la entrada **desire.**

**de·sire**[1] dɪ'zaɪə· r |Ws6| **1** |T1.3,5c:V3| *fml* to wish or want very much: *I desire happiness.*|*I desire to be happy. The Queen desires that you (should) come at once. She desires you to come at once.*|*Give our guests whatever they desire* **2** |T1| to wish to have sexual relations with

**desire**[2] *n* **1** |C:U: (*for*, 3,5c)| a strong wish: *I am filled with desire to go back there.*|*He has a strong desire to succeed*|*for success.*|*many unsatisfied desires*|*his desire that you should do it* **2** |C:U: (*for*)| a strong wish for sexual relations with: *Antony's desire for Cleopatra* **3** |C (*for*), C3,5| an expressed wish or order: *I shall try to act according to your desires.*|*He expressed a desire to see the papers* **4** |C9 *usu. sing.*| something or someone desired: *What is your greatest desire*|*your heart's desire?*

USAGE One can feel **desire** for anything. **Appetite** is only for things of the body, esp. food, and **lust** (*derog*) is a very strong word, particularly for sex.

Entrada desire en Longman

**desire** [dɪ'zaɪə*] **1** n deseo m (*for* de, *to* + infin de + infin); **sexual** — instinto m sexual; **I haven't the least** — **to go** no tengo el menor deseo de ir; **to meet someone's** — satisfacer los deseos de uno.

**2** vt **(a)** desear; querer tener; **to** — **to do** desear hacer; **what does madam** —**?** ¿qué manda la señora?

**(b) to** — **someone to do something** (*wish*) rogar a uno hacer algo, (*order*) mandar a uno hacer algo.

Entrada desire en Collins

[10] No es propósito de esta comunicación entrar en detalles sobre el desarrollo y resultados de este proyecto, pero sí es interesante mencionarlo como un recurso más de extracción automática de información a partir de diccionarios electrónicos.

En Longman, **desire** tiene dos homógrafos de los cuales vamos a tener en cuenta la codificación del primero (como verbo) únicamente. Como verbo, tiene dos sentidos: el primero viene marcado con los códigos gramaticales [T1,3,5c;V3], y el segundo con el código [T1]:

- **T1** significa que **desire** tiene un uso transitivo en el que el objeto directo es un expresión nominal. Esto se codifica en LMT LEF como **v(obj).** La traducción del *Collins English-Spanish* para este uso es *desear* o la perifrasis *querer tener*, que Collins considera sinónimos.
- **T3** significa que **desire** tiene un uso transitivo en el que el objeto directo es un infinitivo con *to*. Esto se codifica en LMT LEF como **v(obj(inf)).** La traducción del *Collins English-Spanish* para este uso es *desear* (ejemplificado como **to desire to do,** *desear hacer).*
- **T5c** signihca que **desire** tiene un uso transitivo en el que el objeto directo es una oración completiva y el verbo de ésta lleva opcionalmente como verbo auxiliar *should*. Esto se codifica en LMT LEF como **v(obj(thatc)).** El *Collins* no ejemplifica su traduccion aunque se inhere que es *desear*.
- **V3** significa que **desire** tiene un uso transitivo con objeto directo doble en el que la segunda parte es un infinitivo con *to*. Esto se codifica en LMT LEF como **v(obj1.comp1(inf)).** La traducción del *Collins English-Spanish* para este uso es *rogar* (ejemplificado como **to desire someone to do something,** rogar *a uno hacer algo* en el contexto de *wish,* y mandar *a uno hacer algo* en el contexto de *order).*

Simplificando todos estos codigos mediante un proceso de unificacion, llegamos a la declaracion de la entrada **desire:**

```
desire < v(obj(inf|thatc)) < t(desear)
       < v(obj1.comp1(inf))
       < t(txt(obj(np):X.comp(inf).empty.obj(thatc):addright(subj(*):X)) ? mandar | desear).
```

Entrada en LMT LEF para desire

## Conclusiones

Con este trabajo, hemos demostrado que estas herramientas son viables para la extracción masiva de información léxica utilizando diversos diccionarios electrónicos como fuentes de conocimiento. De hecho, se han obtenido de esta forma varios miles de entradas para LMT, cuya consistencia y completitud está siendo cotejada con otras fuentes. Esta tarea de revisión consume incluso más recursos humanos que la primera de diseñar los procedimientos de extracción automática. El propósito final es llegar a obtener un diccionario bilingüe básico en formato LMT LEF de alrededor de 60.000 entradas. Nuestros esfuerzos se centran ahora en esta dirección.

Aprovechando la experiencia adquirida con el prototipo inglés-español, relativa al tratamiento lexicográfico, hemos empezado a trabajar también en la adquisición de información para un prototipo de TA de español-inglés. Los primeros resultados son bastante alentadores pero todavía es necesario más tiempo de desarrollo.

## Referencias

**[Anick/Pustejovsky 90]** Anick, P., y Pustejovsky, J. (1990). "An Application of Lexical Semantics to Knowledge Acquisition from Corpora", en *Proceedings of the 13th International Conference on Computational Linguistics*, vol 2, pp. 7- 12.

**[Boguraev/Briscoe 87]** Boguraev, B., y Briscoe, T. (Eds) *(1987).* "Large Lexicons for Natural Language Processing: Exploring the Grammar Coding System of LDOCE", *Computational Linguistics, 13(3-4)* pp. *203 - 218.*

**[Boguraev/Briscoe 89]** Boguraev, B., y Briscoe, T. (Eds) *(1989). Computational Lexicography for Natural Language Processing.* Longman, Harlow.

**[Boguraev et al. 90]** Boguraev, B.; Byrd, R.; Klavans, J., y Neff, M. (1990). *From Structured Analysis of Lexical Resources to Semantics in a Lexical Knowledge Base .,* IBM Technical Report RC *15427,* IBM Thomas J. Watson Research Center, Yorktown Heights, New York.

**[Byrd et al. 87]** Byrd, R.; Calzolari, N.; Chodorow, M.; Klavans, J.; Neff, M., y Rizk, O. *(1987).* "Tools and Methods for Computational Lexicologyn, en *Computational Linguistics,* vol. *13(3-4),* pp. *219 - 240.*

**[Byrd 89]** Byrd, R. *(1989). LQL User Notes: An Informal Guide fo the Lexical Query Language.* IBM Research Report RC *14853,* IBM Thomas J. Watson Research Center, Yorktown Heights, New York.

**[Calzolari/Picchi 86]** Calzolari, N., y Picchi, E. *(1986)."* A Project for a Bilingual Lexical Database System" en *Advances in Lexicology.* Second Annual Conference of the University of Waterloo Centre for the New Oxford English Dictionary, pp. *79 - 92.*

**[Calzolari/Picchi 88]** Calzolari, N., y Picchi, E. (1988). "Acquisition of Semantic Information from an On-Line Dictionaryn, en *Proceedings of the 12th International Conference on Computational Linguistics, pp. 87 - 92.*

**[Calzolari/Bindi 90]** Calzolari, N., y Bindi, R. (1990). "Acquisition of Lexical Information from a Large Textual Italian Corpus", en *Proceedings* of *the 13th International Conference on Computational Linguistics,* vol *3,* pp. *54 - 59.*

**[Cobuild 87]** *Colling Birmingham University International Language Database* (1987). Collins, Glasgow.

**[Collins 79]** *Collins Spanish-English English-Spanish Dictionary* (1979). Ediciones Grijalbo, Barcelona.

**[Gruber 65]** Gruber, J. (1965). *Studies in Lexical Relations.* MIT PhD.

**[Jackendoff 72]** Jackendoff, R. (1972). *Semantic Interpretation in Generative Grammar.* MIT Press, Cambridge.

**[Klavans et al. 91a]** Klavans, J.; Byrd, R.; Chodorow, M., y Wacholder, N. *(1991).* nTaxonomy and Polysemyn, IBM Internal Report RC 16443.

**[Klavans et al. 91b]** Klavans, J.; Chodorow, M., y Wacholder, N. (1991). "From Dictionary to Knowledge Base via Taxonomy", IBM Internal Report RC 16464.

**[Larousse 84]** *Gran Diccionario Espanol-Ingles English-Spanish* (1984). Larousse, Méjico D.F.

**[Longman 81]** *Longman Dictionary of Contemporary English (1981)*. Longman, London.

**[McCord 87]** McCord, M. C. (1987). "Natural Language Processing and Prolog", en Adrian Walker, Michael McCord, John Sowa y Walter Wilson (Eds) *Knowledge Systems and Prolog*. Addison-Wesley, Waltham, Mass, pp. 291- 402.

**[McCord 89a]** McCord, M. C. (1989). "A New Version of Slot Grammarn Research Report RC 14506, IBM Research, Yorktown Heights.

**[McCord 89b]** McCord, M. C. (1989). "Design of LMT: A Prolog-Based Machine Translation System" *Computational Linguistics* 15, pp. 33-52.

**[McCord 89c]** McCord, M. C. (1989). "LMT" en *Proceedings of the MT Summit II*, Munich.

**[McCord 89d]** McCord, M. C. (1989). "A New Version of the Machine Translation System LMT" *Literary and Linguistic Computing* vol. 15, pp. 218-229.

**[McCord 90]** McCord, M. C. (1990). "Slot Grammar: A System for Simpler Construction of Practical Natural Language Grammars" en R. Studer (ed.), *Natural Language and Logic: International Scientific Symposium*, Lecture Notes in Computer Science, Springer Verlag, Berlin.

**[McCord/Wolff 90]** McCord M. C., y Wolff, S. (1990). "The Lexicon and Morphology for LMT, a Prolog-Based MT System" IBM Internal Report RC 59931.

**[McCord/Schwall 90]** McCord, M. C., y Schwall, U. (1990). XSG I *LMT System Usage and Lexical Formalism*. IBM Technical Report, Yorktown/Heidelberg.

**[Neff et al. 88]** Neff, M.; Byrd, R., y Rizk, O. (1988). "Creating and Querying Hierarchical Lexical Data Bases", en *Proceedings of the Second Conference on Applied Natural Language Processing*, Austin, Texas, pp. 84 - 93.

**[Neff/Boguraev 89]** Neff, M., y Boguraev, B. (1989). "Dictionaries, Dictionary Grammars and Dictionary Entry Parsing", en *Proceedings* of *the 27th Annual Meeting of the Assocratioh for Computational Linguistics*. Vancouver, BC, pp. 91-101.

**[Neff/Boguraev 90]** Neff, M., y Boguraev, B. (1990). *Dictionary Entry Parser Reference Manual*. IBM Technical Report, IBM Thomas J. Watson Research Center, Yorktown Heights, New York.

**[Neff/McCord 90]** Neff, M., y McCord, M. (1990). "Acquiring Lexical Data from Machine Readable Dictionary Resources for Machine Translation", en *Proceedings of the 3rd International Conference on Theoretical and Methodological Issues in Machine Translation*, Austin, TX.

**[Quirk et al. 85]** Quirk, R.; Greenbaum, S,; Leech, G., y Svartvik, J. (1985). *A Comprehensive Grammar of the English Language*. Longman.

**[Tesniere 59]** Tesniere, L. (1959). *Elements de Syntaxe Structurale*. Klincksieck, Paris.

**[Webster 87]** *Webster's Seventh New Collegiate Dictionary* (1987). C.&C. Merrian Company, Springfield, Mass.