PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE

FACULTAD DE LETRAS

# Introducción a la lingüística computacional


**César Antonio Aguilar**

**Facultad de Lenguas y Letras**


**21/09/2017**


Cesar.Aguilar72@gmail.com

# NLTK y etiquetado (1)

Vamos a seguir en esta clase con la exploración de corpora en NLTK, de tal suerte que podamos un panorama amplio de lo que podemos explotar usando estas colecciones de corpora, tanto para el inglés como para el español.

Ahora bien, para hacer esta clase de pruebas, necesitamos un módulo nuevo de apoyo, que se llama *operative system*, **OS**. La instrucción para llamarlo es:

```
import os
```

# NLTK y etiquetado (2)

A continuación, visitemos a un viejo conocido:

nltk.corpus.brown.tagged_words()

El resultado, como se darán cuenta, es la oración inicial del sub-corpus *News* del Brown:

[('The', 'AT'), ('Fulton', 'NP-TL'), ('County', 'NN-TL'), ...]

# NLTK y etiquetado (3)

Otro corpus etiquetado muy recurrido parta estas tareas es el **Treebank**, el cual ha establecido una serie de estándares útiles para diseñar anotaciones morfosintácticas. Podemos acceder a él del siguiente modo:

```
nltk.corpus.treebank.tagged_words()
```

Al igual que el Brown, la muestra es un fragmento de una oración:

```
[('Pierre', 'NNP'), ('Vinken', 'NNP'), (',', ','), ...]
```

# NLTK y etiquetado (4)

Vamos a implementar un proceso de etiquetado que considere:

1. La inserción de un archivo, el cual vamos a transformar en una cadena de caracteres.

2. Posteriormente, lo dividimos en palabras.

3. Una vez convertido en lista de palabras, insertamos etiquetas en tal documento.

4. Presentar los resultados en un nuevo texto plano, en donde las palabras van asociadas a una etiqueta correspondiente.

# NLTK y etiquetado (5)

## 1. Importando archivo:

```
Cadena_Turing01 = open(‘c://python27/Tareas_PLN/turing01.txt', 'rU').read()
```

## 2. Comprobamos:

```
type(Cadena_Turing01)

len(Cadena_Turing01)
```

# NLTK y etiquetado (6)

3. Seccionemos nuestra cadena en palabras, usando la función **word_tokenize**:

```
Tokens_Turing01 = nltk.word_tokenize(Cadena_Turing01)
```

4. Comprobamos:

```
type(Tokens_Turing01)

len(Tokens_Turing01)
```

# NLTK y etiquetado (7)

5. Ahora, vamos a añadir etiquetas morfosintácticas a esta lista de tokens, usando la función **pos_tag**, la cual importa tales etiquetas desde el corpus Brown:

```
Tagged_Turing01 = nltk.pos_tag(Tokens_Turing01)
```

4. Comprobamos:

```
type(Tagged_Turing01)

len(Tagged_Turing01)

Tagged_Turing01[0:101]
```

# NLTK y etiquetado (8)

6. Para concluir, vamos a escribir un código que nos permita imprimir nuestros resultados en un archivo de texto plano. Así, lo que hacemos es:

```
Final_Turing01 =
open((‘c://python27/Tareas_PLN/Resultados_Turing01.txt', ‘w')
for elem in Tagged_Turing01:
        Final_Turing01.write(str(elem)+" ")

Final_Turing01.close()
```

Vamos a explicar este código. Lo que estamos pidiendo es: *en un nuevo archivo llamado **Final_Turing01**, el cual se imprimirá con el nombre **Resultados_Turing01.txt**, escribe los elementos que aparecen en el archivo **Tagger_Turing01**, transformándolos de una cadena a una lista de objetos. Al final, cierra el archivo **Final_Turing01**.*

# NLTK y etiquetado (9)

## Y el resultado es:

('THE', 'DT') ('DEVELOPMENT', 'NNP') ('OF', 'NNP') ('PERSON', 'NNP') ('AGREEMENT', 'NNP') ('MARKERS', 'NNP') (':', ':') ('FROM', 'IN') ('PRONOUNS', 'NNPS') ('TO', 'TO') ('HIGHER', 'NNP') ('ACCESSIBILITY', 'NNP') ('MARKERS', 'NNP') ('*', '-NONE-') ('Mira', 'NNP') ('Ariel', 'NNP') ('(', 'NNP') ('2000', 'CD') (')', 'CD') ('Tel-Aviv', 'NN') ('University', 'NNP') ('1', 'CD') ('.', '.') ('From', 'IN') ('free', 'JJ') ('pronouns', 'NNS') ('to', 'TO') ('verbal', 'JJ') ('agreement', 'NN') (':', ':') ('Introduction', 'NNP') ('Grammatical', 'NNP') ('items', 'NNS') ('often', 'RB') ('begin', 'VBP') ('their', 'PRP$') ('linguistic', 'JJ') ('life', 'NN') ('as', 'IN') ('regular', 'JJ') ('lexical', 'JJ') ('items', 'NNS') ('(', ':') ('Meillet', 'NNP') ('1912', 'CD') (')', 'CD') ('.', '.') ('Grammaticization', 'NNP') ('is', 'VBZ') ('said', 'VBN') ('to', 'TO') ('have', 'VB') ('occurred', 'VBN') ('when', 'WRB') ('the', 'DT') ('position', 'NN') ('of', 'IN') ('such', 'JJ') ('lexemes', 'NNS') ('(', ':') ('or', 'CC') ('even', 'RB') ('phrases', 'VBZ') (')', ':') ('becomes', 'NNS') ('fixed', 'VBN') (',', ',') ('their', 'PRP$') ('meaning', 'NN') ('is', 'VBZ') ('generalized/bleached', 'JJ') (',', ',') ('their', 'PRP$') ('domain', 'NN') ('of', 'IN') ('applicability', 'NN') ('enlarged', 'VBD') (',', ',') ('significantly', 'RB') ('raising', 'VBG') ('their', 'PRP$') ('frequency', 'NN') (',', ',') ('their', 'PRP$') ('form', 'NN') ('modified', 'VBD') ('(', 'CD') ('usually', 'RB') ('phonetically', 'RB') ('reduced', 'VBN') (')', '-NONE-') (',', ',') ('and', 'CC') ('their', 'PRP$') ('occurrence', 'NN') ('made', 'VBD') ('obligatory', 'NN') (',', ',') ('even', 'RB') ('when', 'WRB') ('informationally', 'RB') ('redundant', 'JJ') ('(', 'NN') ('see', 'NN') ('Bybee', 'NNP') ('et', 'NN') ('al', 'NN') ('.', '.') ('1994', 'CD') (')', 'CD') ('.', '.') ('Bybee', 'NNP') ('(', 'NNP') ('1985', 'CD') (':', ':') ('38', 'CD') (')', 'CD') ('described', 'VBN') ('the', 'DT') ('process', 'NN') (',', ',') ('saying', 'VBG') ('that', 'IN') ("", 'NN') ('frequent', 'NN') ('items', 'NNS') ('are', 'VBP') ('gradually', 'RB') ('reduced', 'VBN') ('both', 'DT') ('phonologically', 'RB') ('and', 'CC') ('semantically', 'RB') (',', ',') ('and', 'CC') ('are', 'VBP') ('simultaneously', 'RB') ('gradually', 'RB') ('fused', 'VBN') (',', ',') ('again', 'RB') ('both', 'DT') ('phonologically', 'RB') ('and', 'CC') ('semantically', 'RB') (',', ',') ('with', 'IN') ('lexical', 'JJ') ('matter', 'NN') ('contiguous', 'JJ') ('in', 'IN') ('the', 'DT') ('syntactic', 'JJ') ('string', 'NN') ("", ':') ('.', '.') ('High', 'JJ') ('frequency', 'NN') ('is', 'VBZ') ('then', 'RB') ('a', 'DT') ('prerequisite', 'NN') ('for', 'IN') ('grammaticization', 'NN') ('(', ':') ('see', 'NN') ('also', 'RB') ('Hopper', 'NNP') ('and', 'CC') ('Traugott', 'NNP') ('1993', 'CD') (')', 'CD') ('.', '.') ('A', 'DT') ('similar', 'JJ') ('process', 'NN') ('of', 'IN') ('grammaticization', 'NN') ('may', 'MD') ('also', 'RB') ('turn', 'VB') ('less', 'JJR') ('grammatical', 'JJ') ('items',

# NLTK y otros etiquetadores (1)

Así visto, parece que la cuestión de anotar de manera automática un documento o un corpus es muy sencilla, y sí, digamos que de algún modo el uso de estas herramientas debiera facilitarnos ese trabajo.

El etiquetador de NLTK no es el único que hay: por lo menos podemos mencionar otros dos: un viejo conocido, el **TreeTagger**, y uno más reciente, el **Freeling**.

TreeTagger

www.ims.uni-stuttgart.de/projekte/corplex/TreeTagger/DecisionTreeTagger.html

FreeLing 2.2

AN OPEN SOURCE SUITE OF LANGUAGE ANALYZERS

http://nlp.lsi.upc.edu/freeling/

# NLTK y otros etiquetadores (2)

Consideremos los resultados que nos arroja el TreeTagger: como saben, este etiquetador nos muestra un tripleta: palabra/etiqueta/lema:

| word | pos | lemma |
|---|---|---|
| The | DT | the |
| TreeTagger | NP | TreeTagger |
| is | VBZ | be |
| easy | JJ | easy |
| to | TO | to |
| use | VB | use |
| . | SENT | . |

¿Qué tan bueno o malo es esta clase de etiquetado en comparación con el de NLTK? Y más importante: ¿cuál de los dos etiquetadores comete menos errores?

Una buena tarea sería precisamente elaborar un método estadístico y lingüístico, que permita evaluar cuáles son las ventajas y desventajas de cada uno de estos dos etiquetadores.

# NLTK para el español: esto es lo que hay (1)

Siendo demasiado francos, si bien NLTK cuenta con un corpus en español, las colecciones de árboles sintácticos llamadas CESS Treebanks (que son derivados del proyecto Freeling), en español y catalán, estos recursos **no permiten todavía desarrollar un proceso de etiquetado de nuevos documentos, como pasa con el corpus Brown.** Las razones son varias:

1. Los corpus CESS están diseñados para mostrar resultados respecto a las consultas que hagamos en ellos, **pero no permiten implementar algún proceso de aprendizaje para un etiquetador en español.**

2. Todavía no se ha desarrollado algún método en NLTK para entrenar un etiquetador en español.

3. No hay un corpus etiquetado de referencia en español, el cual sea de libre uso, y pueda formar parte de las librerías de NLTK.

4. No hay criterios unificados sobre el proceso de etiquetado de palabras en español.

# NLTK para el español: esto es lo que hay (2)

Si probamos la *demo* que ofrece Freeling en línea, podemos ver que se cuentan con algunos recursos para el etiquetado de oraciones, p.e.:

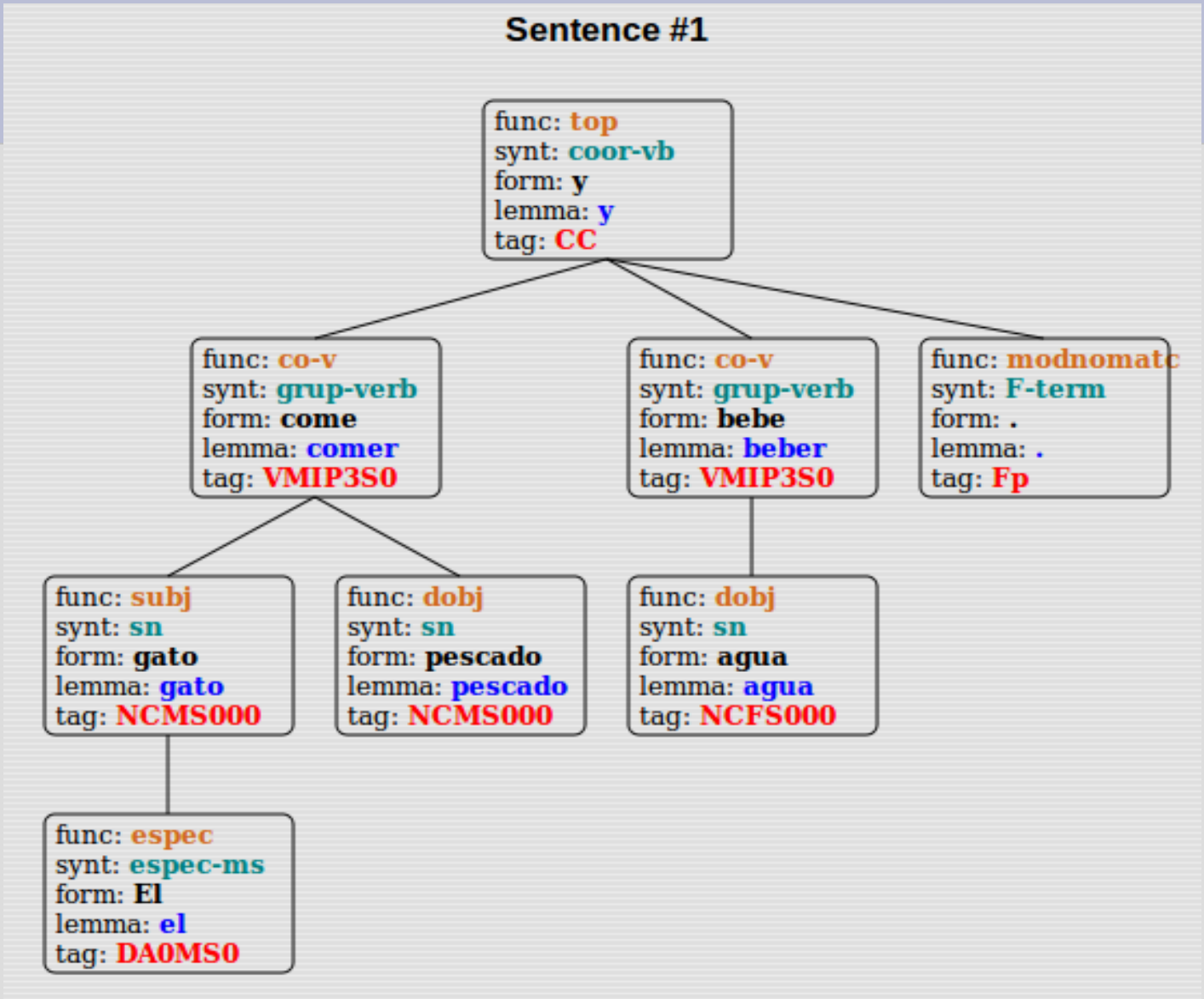
**Analysis Results**

**Sentence #1**

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| **El** | **gato** | **come** | **pescado** | **y** | **bebe** | **agua** | **.** |
| *el* | *gato* | *comer* | *pescado* | *y* | *beber* | *agua* | *.* |
| DA0MS0 | NCMS000 | VMIP3S0 | NCMS000 | CC | VMIP3S0 | NCFS000 | Fp |

Incluso podemos graficar árboles sintácticos con cierto grado de complejidad, digamos un *parseo* basado en gramática de dependencias. Veamos la siguiente lámina:

# NLTK para el español: esto es lo que hay (3)



# NLTK para el español: esto es lo que hay (4)

Esto último sí podemos hacerlo desde NLTK: lanzar consultas al corpus CESS en español, mostrar sus resultados, e incluso generar árboles sintácticos con las oraciones que hay en tal corpus. Así, si ocupan la instrucción:

```
from nltk.corpus import cess_esp

nltk.corpus.cess_esp.words()
```

Nos muestra el ejemplo de una oración:

```
['El', 'grupo', 'estatal', 'Electricit\xe9_de_France', ...]
```

# NLTK para el español: esto es lo que hay (5)

Las etiquetas que ocupa este corpus son una variante del sistema de anotación EAGLES,el cual ustedes ya conocen. Si escriben:

```
nltk.corpus.cess_esp.tagged_words()
```

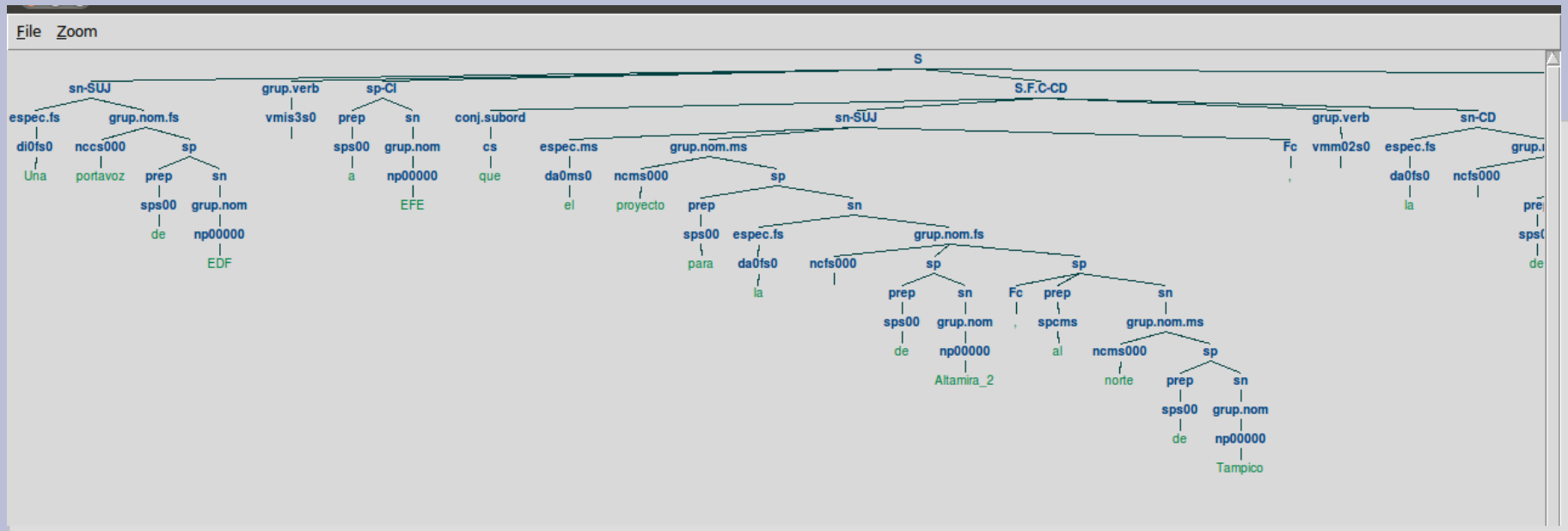
Nos arroja como resultado:

```
[('El', 'da0ms0'), ('grupo', 'ncms000'), ...]
```

Finalmente, podemos generar un esquema arbóreo, seleccionado alguna de las oraciones que cuenta este corpus. Usemos esta instrucción, y veamos el resultado en la siguiente lámina:

```
Tree01 = nltk.corpus.cess_esp.parsed_sents()[1].draw()
```

# NLTK para el español: esto es lo que hay (6)




A manera de comentario final: hay muchas cosas por hacer todavía respecto al uso de NLTK para el procesamiento de corpus, pero lo importante es que hay soluciones. ¿Qué posibles vías creen que puedan ser útiles para aprovechar estos recursos para el análisis de textos en español?

# FreeLing hoy en día (1)

Cabe señalar que el proyecto que más ha avanzado en español hoy en día es **FreeLing**. Hoy en día, este sistema de etiquetado cuenta con varios recursos aplicables no solo para el español, sino para otras variantes dialectales y lenguas.

- The Asturian dictionary contains more than 140,000 forms corresponding to some 40,000 combinations lemma-PoS.
- The Catalan dictionary contains more than 520,000 forms corresponding to 71,000 combinations lemma-PoS.
- The Welsh dictionary contains some 345,000 forms, corresponding to over 8,500 lemma-PoS combinations.
- The German dictionary contains near 395,000 forms, corresponding to almost 130,000 lemma-PoS combinations.
- The English dictionary was automatically extracted from WSJ and other corpuses, with accurate manual post-edition and completion. It contains about 68,000 forms corresponding to some 37,000 different combinations lemma-PoS.
- The Spanish dictionary contains over 555,000 forms corresponding to more than 76.000 lemma-PoS combinations.
- The French dictionary contains over 350,000 forms corresponding to more than 54.000 lemma-PoS combinations.
- The Galician dictionary contains some 430,000 forms, corresponding to near 50,000 lemma-PoS combinations.
- The Italian dictionary contains over 360,000 forms corresponding to 40,000 lemma-PoS combinations.
- The Norwegian dictionary contains 618,000 forms corresponding to 145,000 lemma-PoS combinations.
- The Portuguese dictionary contains about 909,000 forms corresponding to 105,000 lemma-PoS combinations.
- The Russian dictionary contains over 1,630,000 forms corresponding to more than 510,000 lemma-PoS combinations.
- The Slovenian dictionary contains about 920,000 forms corresponding to some 94,000 lemma-PoS combinations.

# FreeLing hoy en día (2)

Empleando las *demos* en línea que tiene FreeLing, podemos ver cómo se relaciona el etiquetado gramatical con el análisis morfológico y sintáctico, p.e.:

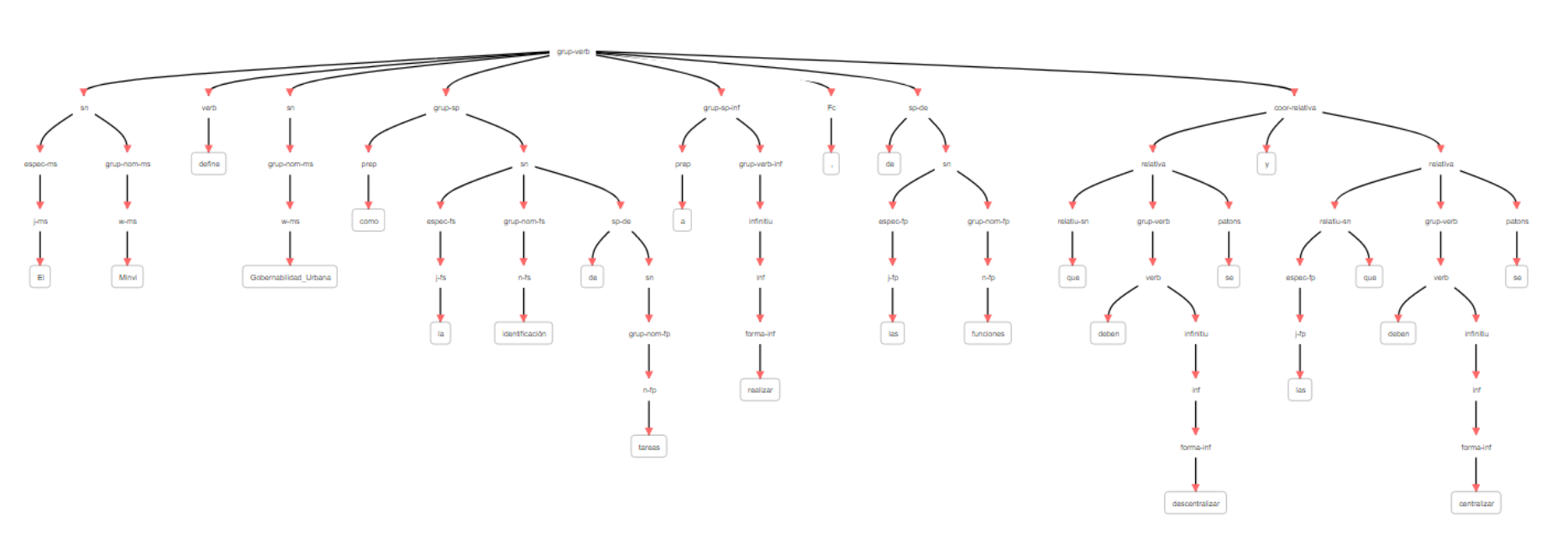
▼ **Sentences**

**Sentence 1**

| El | Minvi | define | Gobernabilidad_Urbana | como | la | identificación | de | tareas | a | realizar |
|---|---|---|---|---|---|---|---|---|---|---|
| el *DA0MS0* 1 | minvi *NP00000* 1 | definir *VMIP3S0* 0.989241 | gobernabilidad_urbana *NP00000* 1 | como *CS* 0.967153 | el *DA0FS0* 0.98926 | identificación *NCFS000* 1 | de *SP* 0.999961 | tarea *NCFP000* 1 | a *SP* 0.998775 | realizar *VMN0000* 1 |
| | | definir *VMM02S0* 0.0107595 | | como *SP* 0.0211062 | lo *PP3FSA0* 0.010734 | | de *NCFS000* 3.85246e-05 | | a *NCFS000* 0.0012246 | |
| | | | | como *PR00000* 0.00821174 | la *NCMS000* 6.20105e-06 | | | | | |
| | | | | como *CC* 0.00346115 | | | | | | |
| | | | | comer *VMIP1S0* 6.78656e-05 | | | | | | |

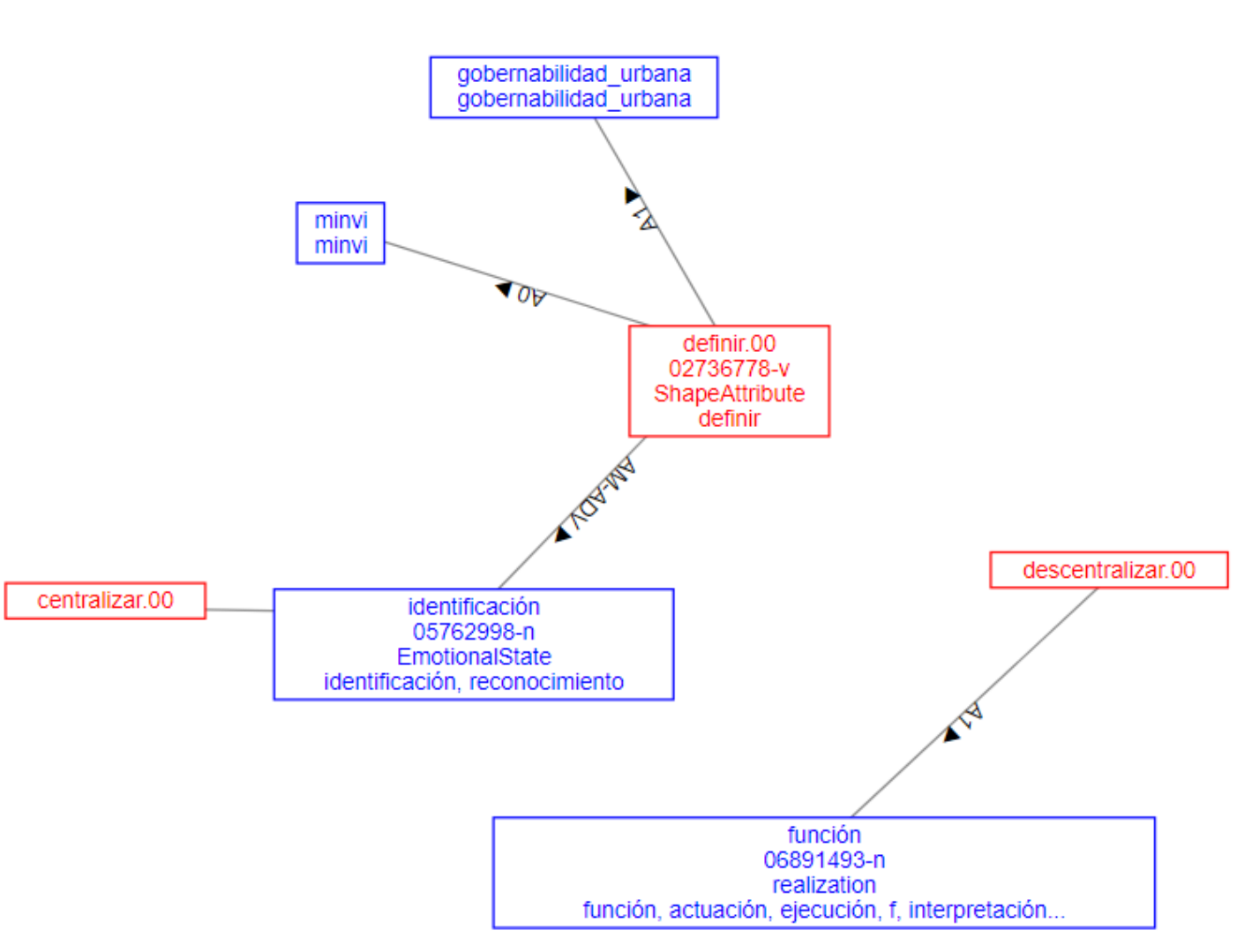
# FreeLing hoy en día (3)

## También permite la generación de árboles sintácticos, p. e.:



# FreeLing hoy en día (4)

E incluso la generación de grafos con información semántica:



# Gracias por su atención

**Blog del curso:**
http://cesaraguilar.weebly.com/introduccioacuten-a-la-linguumliacutestica-computacional.html