Eines i recursos per al català i castellà

Antoni Oliver

PID_00155980

Universitat Oberta de Catalunya
www.uoc.edu

Índex

Introducció

En els mòduls anteriors hem vist diferents tasques relacionades amb el processament del llenguatge natural i hem après a crear programes en Python i NLTK que portaven a terme les tasques estudiades. Les aplicacions de processament del llenguatge natural necessiten una gran quantitat de dades lingüístiques (formaris, diccionaris bilingües, *wordnets*, etc.). Algunes d'aquestes dades estan disponibles com a part integrant d'aplicacions de programari lliure i és possible fer-les servir en les nostres aplicacions sempre que respectem els termes de la llicència. En aquest mòdul veurem una sèrie d'eines de programari lliure que permeten portar a terme diferents tasques de processament del llenguatge natural i a la vegada aprofitar les dades lingüístiques que proporcionen. De les eines disponibles hem triat les que estan disponibles per al català i, com veurem, moltes estan disponibles per al castellà i altres llengües.

En primer lloc presentem El Corrector, un corrector ortogràfic i gramatical per al català. Presentarem aquesta aplicació molt per sobre, ja que el que ens interessarà per a aquest mòdul és el seu *formari*, és a dir, el seu diccionari de formes catalanes, que com veurem és molt complet.

Posteriorment presentarem DACCO, un diccionari català-anglès lliure, que ens servirà per a veure com en podem processar les entrades amb Python.

Com a analitzador lingüístic presentem el Freeling. Aprendrem a instal·lar-lo tant en Linux com en Windows i a accedir a les dades lingüístiques que proporciona.

I finalment presentarem un sistema de traducció automàtica de programari lliure, l'Apertium. Malauradament ara per ara només es pot instal·lar sota Linux. Explicarem com es fa la instal·lació sota aquest sistema operatiu i quines dades lingüístiques proporciona. També veurem com podem processar aquestes dades lingüístiques amb Pyhton.

Objectius

Els objectius bàsics que ha d'haver aconseguit l'estudiant una vegada treballats els continguts d'aquest mòdul són els següents:

1. Saber instal·lar i fer servir l'analitzador Freeling.
2. Conèixer el sistema de traducció automàtica Apertium.

1. El Corrector

El Corrector (Martí Quixal i Valentín, 2008) és un corrector ortogràfic i gramatical de la llengua catalana, de codi obert, que funciona sota diverses plataformes. Ha estat desenvolupat pel Grup de Lingüística Computacional de la Universitat Pompeu Fabra, amb la col·laboració de Barcelona Media, i en gran part és el resultat del finançament rebut a partir d'un concurs públic convocat per la Generalitat de Catalunya.

Adreça web recomanada

Es pot obtenir informació addicional sobre El Corrector a la plana web http://www.elcorrector.cat.

Com ja sabeu, per a desenvolupar un corrector ortogràfic cal disposar d'una llista de formes correctes de la llengua, és a dir, una llista de paraules que contingui totes les formes flexionades de "totes" les paraules de la llengua. Com que no és possible disposar de totes les paraules d'una llengua, interessa que aquesta llista sigui el més àmplia possible. Si a més el corrector és també gramatical necessitarà certa informació gramatical de cada una d'aquestes formes. Sovint aquestes llistes de paraules flexionades reben el nom de *formaris* i si van acompanyades del lema i d'informació morfosintàctica reben el nom de *diccionaris morfològics*.

Com que el projecte de desenvolupament d'*El Corrector* és de codi obert podrem accedir també a les dades lingüístiques. En concret el que farem serà obtenir aquest formari i veure quina informació conté.

Per a poder accedir al codi font d'El Corrector cal anar a l'adreça web següent: https://svn.projectes.lafarga.cat/svn/corrector. El sistema demanarà un nom d'usuari i contrasenya; cal introduir el nom d'usuari *anonymous* sense cap contrasenya. Un cop dins hem d'anar a la següent secció de la web /Codi Font/LingResources/Catalan/ElCorrector/tags/1.0/Data i baixar el fitxer `MainDictCore.dat`. Aquest fitxer és de text i té l'aspecte següent:

```
alarmat#alarmar:VC--SM:0:1:1508082:LP
alarmats#alarmar:VC--PM:0:1:8545798:LP
alarmau#alarmar:VDR2P-:3:1:0:LP
alarmau#alarmar:VRR2P-:3:1:0:LP
alarmava#alarmar:VDA1S-:0:1:51274788:LP
alarmava#alarmar:VDA3S-:0:1:3016164:LP
alarmaven#alarmar:VDA3P-:0:1:17091596:LP
alarmaves#alarmar:VDA2S-:0:1:0:LP
alarme#alarmar:VDR1S-:4:1:0:LP
```

El fitxer conté 901.129 entrades, una per línia. Fixem-nos que a cada línia hi ha una forma seguida d'un signe "#" i tot d'informació. La primera informació que trobem és el lema associat a la forma i després les etiquetes morfosintàctiques. Les categories majors que preveu l'etiquetari emprat es poden observar a la taula 1.

Taula 1. Categories majors emprades per *El Corrector*

Etiqueta	Categoria
C	conjunció
D	adverbi
E	determinant
H	adjectiu/ (verb) participi
I	interjecció
J	adjectiu
L	locució
N	nom
P	preposició
R	pronom
V	verb
W	no analitzable

Podem trobar una descripció detallada dels recursos lingüístics als documents que es troben a /Codi Font/LingResources/Catalan/ElCorrector/tags/1.0/doc/Diccionaris.

Com que aquest formari està en un fitxer de text serà molt senzill obrir-lo amb Python i fer el tractament que necessitem. Per exemple, en el programa següent (`programa-10-1.py`) obrim el diccionari, separem convenientment la informació i desem en un diccionari les diferents formes i la informació morfosintàctica associada.

```
formari=open("MainDictCore.dat",'r')
cont=0
diccionari={}
for linia in formari.readlines():
    cont+=1
    if cont > 3:     #les tres primeres línies contenen altra informació
        info1=linia.rstrip().split("#")
        info2=info1[1].split(":")
        info3=info2[1].split(",")
        forma=info1[0]
        lema=info2[0]
        etiqueta=info3[0]
        if diccionari.has_key(forma):
            diccionari[forma]+=","+lema+":"+etiqueta
        else:
            diccionari[forma]=lema+":"+etiqueta

#fem una consulta
```

```
print diccionari['casa']
```

Si executem el programa ens donarà la sortida següent:

```
casa:N5-FS,casar:VDR3S-,casar:VRR2S-
```

2. DACCO

DACCO és un diccionari anglès-català i català-anglès de codi lliure. A la plana web http://www.catalandictionary.org es pot trobar tota la informació sobre el projecte DACCO. En aquesta plana es pot consultar el diccionari, baixar-lo per a fer consultes directament en el nostre ordinador, baixar versions imprimibles i baixar els diccionaris en format XML. A la figura 1 podem observar l'aspecte de la interfície d'aquest diccionari.

Figura 1. Interfície del diccionari DACCO

El que ens interessarà en aquest cas serà baixar els diccionaris en format XML i ser capaços de processar-los amb Python. Baixem de l'enllaç corresponent l'arxiu font i el descomprimim. Es crearà una estructura de directoris força complexa. Al directori /Dacco-0.9/dictionaries/engcat i al directori Dacco-0.9/dictionaries/cateng són els diccionaris en format XML. Hi ha un fitxer XML per a cada lletra inicial, per exemple, `a.xml`. Copiem tots els arxius amb extensió `.dic` a un directori diferent per a poder-hi treballar. Si ens fixem, els diccionaris tenen l'estructura següent:

```
<Entry frequency="190000»abdicate<verbs>
    <translations>
        <translation intransitive="true»abdicar
            <example>If Queen Elizabeth &lt;b&gt;abdicates&lt;/b&gt;
                        then Prince Charles will become  King.</example>
        </translation>
```

```
        <translation transitive="true»renunciar a
            <example>Even if you find a great paediatrician, you cannot &lt;b&gt;
                    abdicate&lt;/b&gt; responsibility for your childrens'
                    health.</example>
            <example>Edward VIII became the first English monarch
                    to &lt;b&gt;abdica te&lt;/b&gt;
                    the throne voluntarily.</example>
        </translation>
    </translations>
    </verbs>
</Entry>
```

Per a processar arxius XML en Pyhton disposem de diverses possibilitats. En els programes que presentem en aquest mòdul farem ús de SAX (*Simple Api for XML*). Veurem que per a construir els nostres programes que analitzin el contingut d'un fitxer en XML el que farem és construir una classe que faci de manipulador (*handler*) per a aquest tipus de XML. Els manipuladors senzills que programarem tindran quatre mètodes:

1) __init__: serà el mètode que es cridi quan s'inicialitzi un objecte amb aquesta classe. El que farem bàsicament serà inicialitzar el valor d'un conjunt de variables.

2) startElement: serà el mètode que es cridi quan l'analitzador trobi l'etiqueta d'inici d'un element; disposarem del nom de l'etiqueta a *name* i el conjunt d'atributs al diccionari *attrs*. Per a consultar el valor d'un cert atribut haurem de consultar el valor d'aquest diccionari per una clau de valor igual a la de l'atribut.

3) characters: es cridarà aquest mètode quan es trobin dades entre etiquetes. Disposarem del valor d'aquestes dades a la variable *data*.

4) endElement: serà el mètode que es cridi quan l'analitzador trobi l'etiqueta de tancament d'un element, disposarem del nom de l'etiqueta a *name*.

En el `programa-10-2.py` llegim tots els arxius de *diccionary* i fem un parell de consultes. El codi del programa és el següent:

```
import sys, codecs
import xml.sax
import xml.sax.handler
import re

class DaccoHandler(xml.sax.handler.ContentHandler):
    """
    A handler to deal with Dacco dictionary format
```

```
    """
    def __init__(self):
        self.inEntry=False
        self.inTranslation=False
        self.diccionari={}
    def startElement(self,name,attrs):
        if name=="Entry":
            self.inEntry=True
            self.original=""
            self.traduccio=[]
        elif name=="translation":
            self.inTranslation=True
        else:
            self.inTranslation=False
            self.inEntry=False
    def characters(self,data):
        if self.inEntry:
            self.original=data
        if self.inTranslation:
            if re.match("[A-Za-z]", data):
                self.traduccio.append(data)
    def endElement(self,name):
        if name=="Entry":
            self.inEntry=False
            self.diccionari[self.original]=", ".join(self.traduccio)
        elif name=="Translation":
            self.inTranslation=False
        else:
            self.inTranslation=False
    def returnDictionary(self):
        return self.diccionari

parser=xml.sax.make_parser()
handler=DaccoHandler()
parser.setContentHandler(handler)

for fitxer in ("a.dic","b.dic"," c.dic","d.dic","e.dic","f.dic","g.dic","h.dic",
ï.dic","j.dic","k.dic",l·l.dic", "m.dic","n.dic","o.dic","p.dic","q.dic","r.dic",
"s.dic","t.dic"," u.dic","v.dic","w.dic","x.dic","y.dic","z.dic"):
    parser.parse(fitxer)

diccionari=handler.returnDictionary()
print "energy",diccionari["energy"]
print "zigzag",diccionari["zigzag"]
```

Si executem aquest programa ens retorna el següent:

```
energy energia, energètic
zigzag zigzaguejar, ziga-zaga, zig-zag
```

Amb els tres mètodes de control podem saber en tot moment quina informació estem llegint i emmagatzemar-la convenientment (en el nostre cas en un diccionari). Finalment definim també un mètode *returnDictionary* que retorna el diccionari format. Fixeu-vos que només ens quedem amb la informació corresponent a l'original i a les seves possibles traduccions.

En el cos del programa principal enviem al manipulador tots els arxius de diccionari, recuperem el diccionari total format i fem un parell de consultes.

3. Freeling

El Freeling (Carreras i altres, 2004) és un conjunt d'analitzadors lingüístics alliberats amb llicència GPL. Actualment la majoria de característiques dels analitzadors estan disponibles per a castellà, català, gallec, anglès i italià. Amb el Freeling podem dur a terme diverses tasques:

- *Tokenització*
- Divisió de text en frases
- Anàlisi morfològica
- Reconeixement d'entitats multiparaula
- Predicció estadística de la categoria gramatical de paraules desconegudes
- Reconeixement d'entitats amb nom
- Classificació d'entitats amb nom
- Reconeixement de dates, nombres, monedes i magnituds físiques (velocitat, pes, temperatura, densitat, etc.)
- Anàlisi morfosintàctica
- Anàlisi sintàctica superficial
- Anotació de sentits basada en WordNet
- Anàlisi de dependències

En aquest apartat explicarem com es pot obtenir i instal·lar el Freeling i fer alguns tipus d'anàlisis bàsiques. Freeling està dissenyat inicialment per a funcionar en Linux, però pot funcionar també sota Windows. Explicarem per separat el procés d'instal·lació; no cal que llegiu l'apartat d'instal·lació per a un sistema operatiu diferent del que feu servir. Pel que fa a la utilització bàsica, són idèntiques per a Windows i Linux.

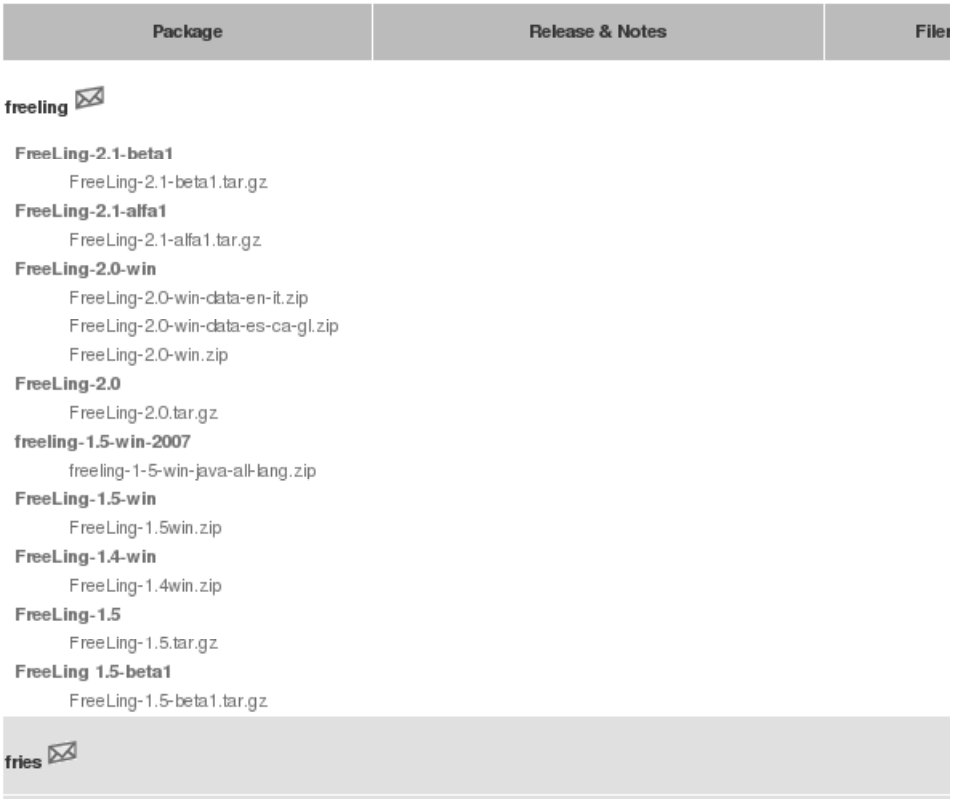
També farem una ullada a les dades lingüístiques que proporciona aquest analitzador i aprendrem a processar-les amb Python.

3.1. Obtenció i instal·lació del Freeling

Podem trobar tots els arxius necessaris per a instal·lar el Freeling en el nostre ordinador en la pàgina web http://garraf.epsevg.upc.es/freeling/, en l'apartat "Downloads". Cal fixar-se en la versió i en el sistema operatiu. Fixeu-vos en la figura 2 (tot i que la plana de baixada pot haver canviat lleugerament).

Convé evitar les versions alfa i beta i anar a una de les estables. Per a Linux caldrà baixar els arxius següents:

Figura 2. Fragment de la web de descàrrega del Freeling



- `Freeling-X.XX.tar.gz` (cal baixar la darrera versió estable)
- `fries-X.XX.tar.gz` (cal baixar la darrera versió estable)
- `omlet-X.XX.tar.gz` (cal baixar la darrera versió estable)

En canvi per a Windows serà necessari baixar els arxius següents:

- `FreeLing-2.0-win.zip`: és el programa en si.

- `FreeLing-2.0-win-data-es-ca-gl.zip`: dades lingüístiques del castellà, català i gallec; baixeu-les únicament si voleu treballar amb aquestes llengües.

- `FreeLing-2.0-win-data-en-it.zip`: dades lingüístiques de l'anglès i italià; baixeu-les únicament si voleu treballar amb aquestes llengües.

3.1.1. Instal·lació sota Linux

Abans d'instal·lar aquests programes, haurem d'instal·lar una sèrie de prerequisits. Tots aquests prerequisits es poden instal·lar directament amb "sudo apt-get install" i el nom del paquet per instal·lar:

- libpcre3
- libpcre3-dev

- libdb4.6++
- libdb4.6++-dev
- libcfg+0
- libcfg+0-dev

Una vegada instal·lats aquests prerequisits passarem a instal·lar el Fries, l'Omlet i finalment el Freeling. Tots aquests programes es poden instal·lar amb la seqüència:

```
sudo gunzip ....tar.gz
sudo tar -xvf ....tar
cd (al directori que s'ha creat)
sudo ./configure
sudo make
sudo make install
```

Si tot ha funcionat correctament, ja tindrem instal·lat el Freeling en el nostre ordinador i el podrem executar mitjançant l'ordre *analyze*.

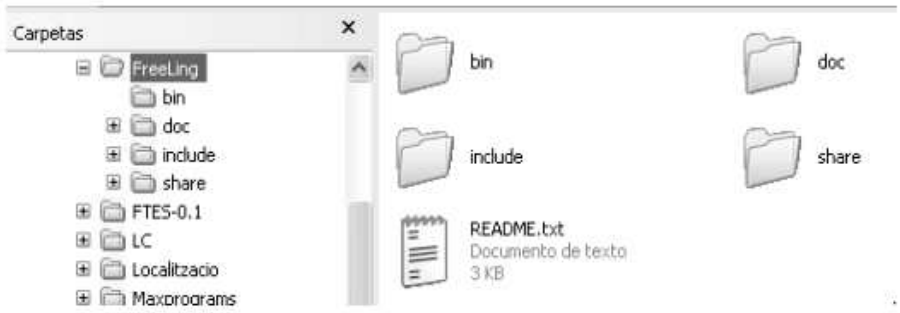
Explicarem l'execució del Freeling en el subapartat 3.2.

3.1.2. Instal·lació sota Windows

Un cop baixats tots els arxius corresponents a la darrera versió disponible de Freeling per a Windows, descomprimiu-los directament a la vostra unitat C. Fixeu-vos que hi ha tres arxius: un és el programa i els altres dos són els corresponents a les dades lingüístiques (un per l'anglès i l'italià, i un altre per al castellà, català i gallec).

Un cop descomprimits els arxius a la vostra unitat C tindreu una carpeta que s'anomena `Freeling` que conté tant els programes com les dades lingüístiques (figura 3).

Figura 3. Carpetes resultants de la instal·lació del Freeling



Per a poder fer servir el programa, encara haurem de fer alguns ajustaments i configuracions, que explicarem a continuació:

1) Establiment del valor de la variable d'entorn `FREELINGSHARE`. Els arxius de configuració del Freeling fan servir una variable d'entorn que s'anomena `FREELINGSHARE`. Per a establir el valor de la variable d'entorn s'han de seguir els passos següents:

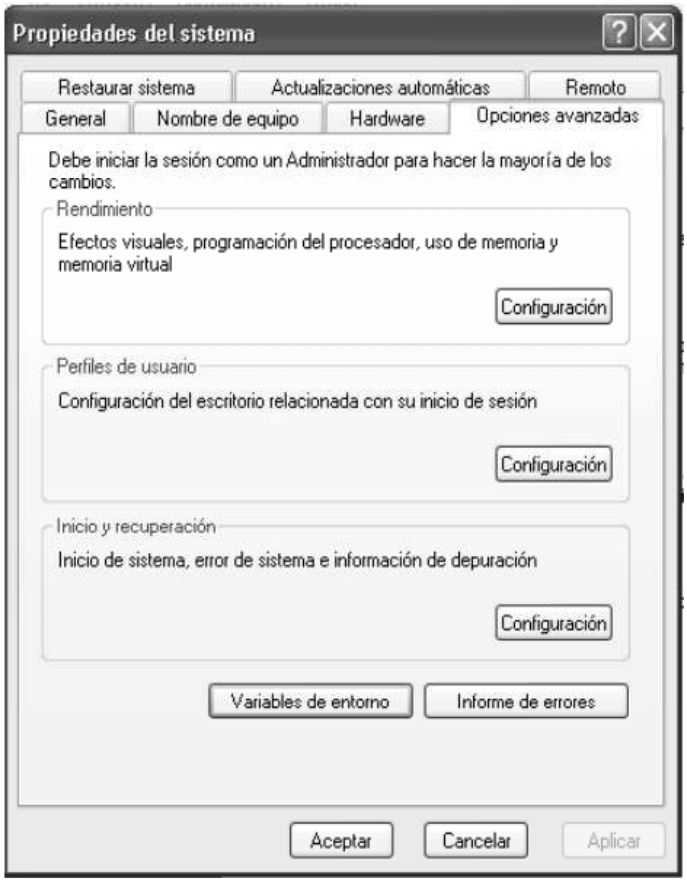
Variable d'entorn

Una variable d'entorn és una variable que queda emmagatzemada en el sistema operatiu i que els programes poden consultar.

a) Aneu a Inici > Panell de Control > Sistema. Atenció: en el Windows Vista només us apareix la informació; caldrà que feu clic en un enllaç on posa "Canvia la configuració".

b) Seleccioneu la pestanya "Opcions avançades" i feu clic al botó "Variables d'entorn" (figura 4).

Figura 4. Opcions avançades



c) Us apareixerà una pantalla com la de la figura 5. Fixeu-vos que si teniu més d'un usuari al vostre ordinador tindreu unes variables assignades a l'usuari actual i unes altres assignades a tot el sistema.

d) Feu clic al botó "Nova" (ho podeu fer a les variables de l'usuari actual). Us apareixerà una pantalla on haureu d'escriure el nom de la variable `FREELINGSHARE` i el valor d'aquesta variable `C:\Freeling\share` (si heu descomprimit els arxius del Freeling directament sobre la unitat C). Observeu la figura 6.

Figura 5. Variables d'entorn

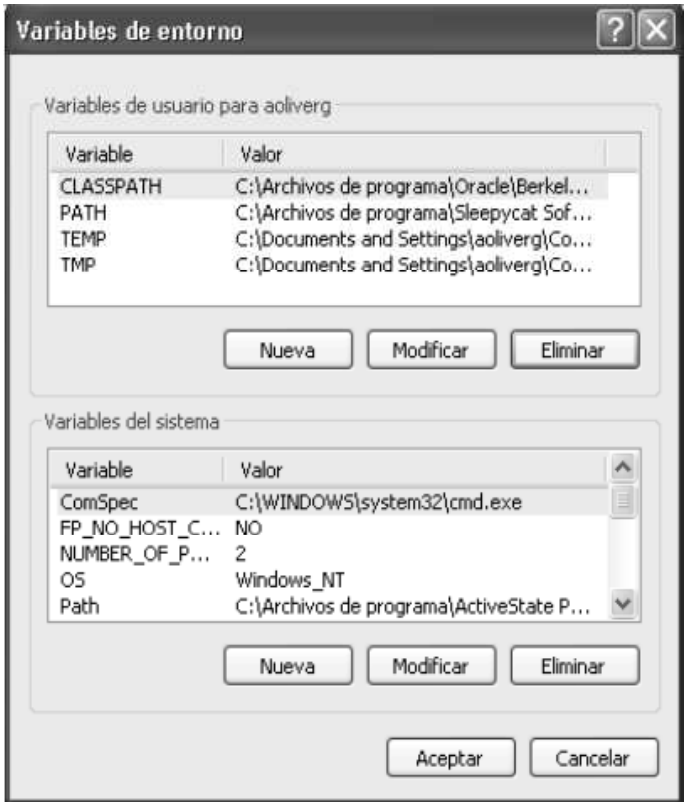


Figura 6. Pantalla de nova variable d'usuari

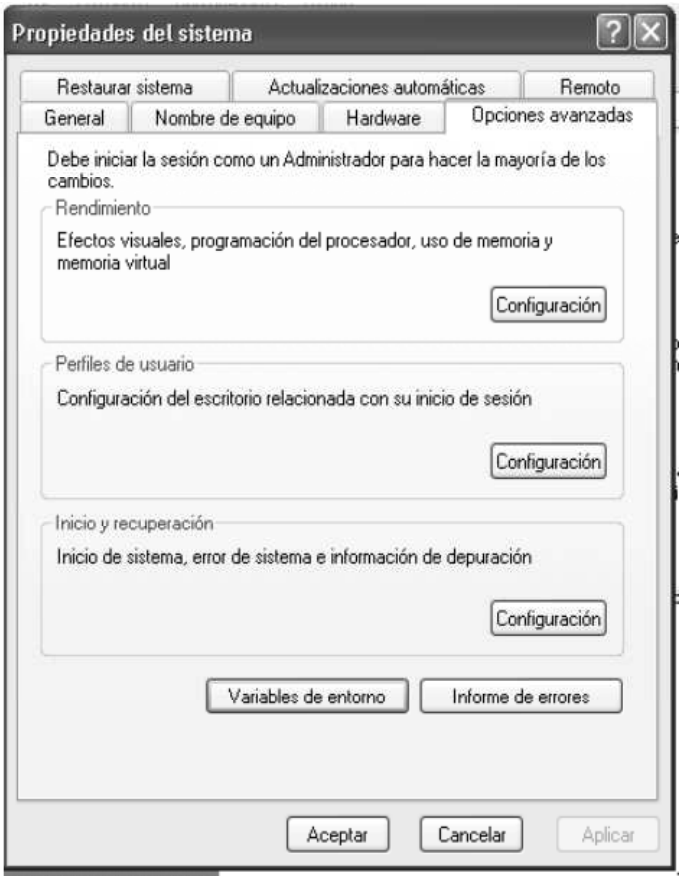
e) Finalment, feu clic sobre el botó "Acceptar" i torneu a fer clic sobre el botó "Acceptar" de la pantalla anterior, i una vegada més per a tancar la pantalla principal de "Sistema".

2) Inclusió del Freeling en el *path* del sistema. Ara ja podríem cridar el Freeling, però cada cop que ho féssim hauríem de posar la ruta completa al programa (`C:\Freeling\bin\analyzer`). Per a evitar això, inclourem el directori `C:\Freeling\bin` al *path* del sistema. Fer això és molt semblant al que hem fet a anteriorment. Fem:

a) Aneu a Inici > Panell de Control > Sistema. Atenció: en el Windows Vista només us apareix la informació; caldrà que feu clic en un enllaç on posa "Canvia la configuració".

b) Seleccioneu la pestanya "Opcions avançades" i feu clic al botó "Variables d'entorn" (figura 7).

Figura 7. Pantalla de propietats del sistema



c) Us apareixerà una pantalla con la que es veu a la figura 8. Fixeu-vos que si teniu més d'un usuari al vostre ordinador tindreu unes variables assignades a l'usuari actual i unes altres assignades a tot el sistema.

d) Ara en "Variables del sistema" busquem i seleccionem "Path" i fem clic al botó "Modificar". Apareix la pantalla de la figura 9.

e) Poseu el cursor sobre el quadre de text "Valor de variable" i aneu al final de tot. Afegiu `;C:\Freeling\bin*`. Si us equivoqueu feu clic a "Cancel·la" i torneu a provar. Si elimineu valors del *path* algunes aplicacions us deixaran de funcionar. Feu clic al botó "Acceptar".

***És molt important *afegir* al final aquest valor i no substituir tot el valor que ja té. No oblideu de posar el ";".**

f) Cliqueu "Acceptar" les vegades necessàries perquè es tanquin les finestres.

3.2. Execució del Freeling

El Freeling necessita que li especifiquem tota una sèrie de paràmetres perquè pugui funcionar. Els podem especificar en la línia d'ordres quan cridem el programa, o bé mitjançant un fitxer de configuració. És molt més convenient utilitzar aquesta segona opció, ja que podem tenir diversos fitxers de configuració preparats per a les diverses llengües i els diferents tipus d'anàlisi que volem fer. En un terminal (Linux) o en la pantalla de "Símbol del sistema" feu:

```
analyzer --help
```

Figura 8. Pantalla variables d'entorn

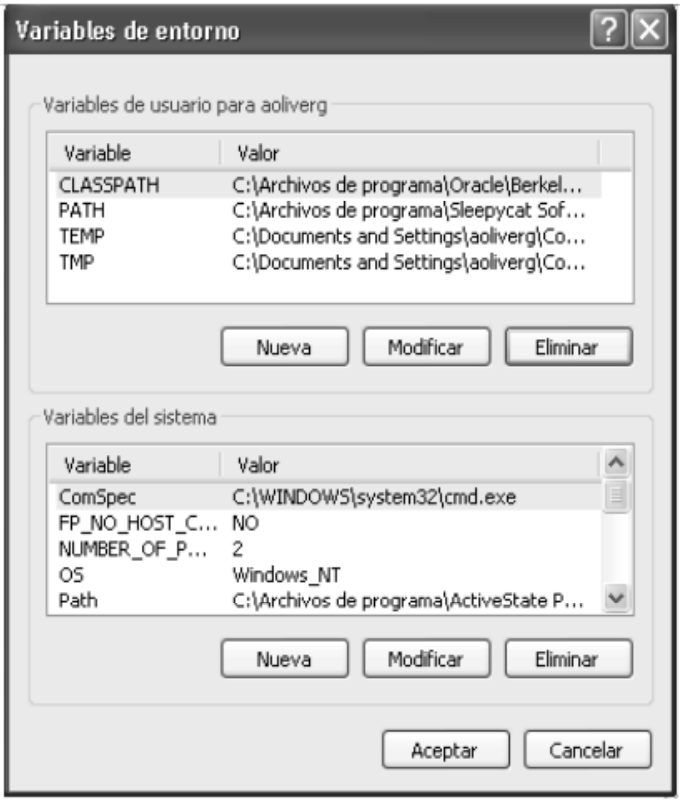
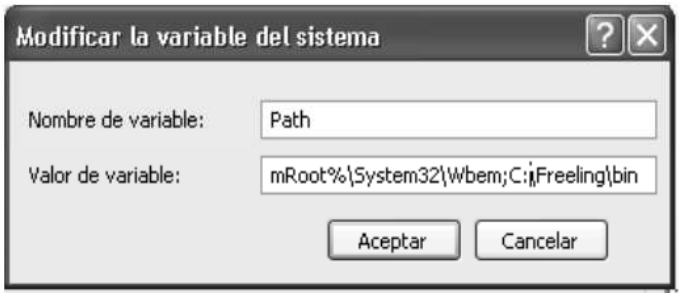


Figura 9. Pantalla de modificació de variable del sistema



Ens mostrarà totes les opcions, i en són moltes:

```
Available options:

-h, --help              This screen
-f filename             Use alternative configuration file (default: analyzer.cfg)
--lang language         Language (sp: Spanish, ca: Catalan, en: English)
--flush                 Consider each newline as a sentence end
--inpf string           Input format (plain,token,splitted,morfo,sense,tagged)
--outf string           Output format (plain,token,splitted,morfo,tagged,parsed,dep)
--ftok filename         Tokenizer rules file
--fsplit filename       Splitter options file
--sufx, --nosufx        Whether to perform suffix analysis
--loc, --noloc          Whether to perform multiword detection
--numb, --nonumb        Whether to perform number detection
```

```
--punt, --nopunt       Whether to perform punctuation detection
--date, --nodate       Whether to perform date and time expressions detection
--quant, --noquant     Whether to perform quantities detection
--dict, --nodict       Whether to perform dictionary search
--prob, --noprob       Whether to perform probability assignment
--ner, --noner         Whether to perform NE recognition
--dec string           Decimal point character
--thou string          Thousand point character
--floc,-L filename     Multiwords file
--fqty,-Q filename     Quantities file
--fsuf,-S filename     Suffix rules file
--fprob,-P filename    Probabilities file
--thres float          Probability threshold for unknown word tags
--title int            Length beyond which an all_caps sentence is considered a
                       title and not a proper noun
--fdict,-D filename    Dictionary database
--fnp,-N filename      NP recognizer data file
--nec, --nonec         Whether to perform NE classification
--fnec filename        Filename prefix for NEC data XX.rgf, XX.lex, XX.abm
--sense string         Type of sense annotation (no|none,all,mfs)
--fsense,-W filename   Sense dictionary file
--dup, --nodup         Whether to duplicate analysis for each different sense
--fpunct,-M filename   Punctuation symbols file
--tag,-T string        Tagging alogrithm to use (hmm, relax)
--hmm,-H filename      Data file for HMM tagger
--rlx,-R filename      Data file for RELAX tagger
--rtk, --nortk         Whether to perform retokenization after PoS tagging
--force string         Whether/when the tagger must be forced to select only one
                       tag per word (none,tagger,retok)
--iter int             Maximum number of iterations allowed for RELAX tagger.
--sf float             Support scale factor for RELAX tagger (affects step size)
--eps float            Epsilon value to decide when RELAX tagger achieves no more changes
--grammar,-G filename  Grammar file for chart parser
--dep,-J filename      Rule file for dependency parser
```

Com veiem, hi ha un gran nombre de paràmetres per especificar i això fa pràcticament inviable utilitzar aquesta opció de crida. Veiem com podem crear aquests fitxers de configuració.

La instal·lació per defecte crea uns fitxers de configuració que podem aprofitar per a les nostres anàlisis. Aquests fitxers de configuració es troben en un sistema Linux a /usr/local/share/Freeling/config i en Windows els trobarem a C:\Freeling\share\config, i n'hi ha un per a cada llengua disponible (`ca.cfg`, `es.cfg`, `en.cfg`, `gl.cfg`, `it.cfg`). Copiem `es.cfg` en el nostre directori i l'obrim amb un editor de textos. Hi veurem el següent:

```
##
#### default configuration file for Spanish analyzer
##

#### General options
Lang=es

#### Trace options. Only effective if we have compiled with -DVERBOSE
#
## Possible values for TraceModule (may be OR'ed)
#define SPLIT_TRACE         0x00000001
#define TOKEN_TRACE         0x00000002
#define MACO_TRACE          0x00000004
#define OPTIONS_TRACE       0x00000008
#define NUMBERS_TRACE       0x00000010
#define DATES_TRACE         0x00000020
#define PUNCT_TRACE         0x00000040
#define DICT_TRACE          0x00000080
#define SUFF_TRACE          0x00000100
#define LOCUT_TRACE         0x00000200
#define NP_TRACE            0x00000400
#define PROB_TRACE          0x00000800
#define QUANT_TRACE         0x00001000
#define NEC_TRACE           0x00002000
#define AUTOMAT_TRACE       0x00004000
#define TAGGER_TRACE        0x00008000
#define HMM_TRACE           0x00010000
#define RELAX_TRACE         0x00020000
#define RELAX_TAGGER_TRACE  0x00040000
#define CONST_GRAMMAR_TRACE 0x00080000
#define SENSES_TRACE        0x00100000
#define CHART_TRACE         0x00200000
#define GRAMMAR_TRACE       0x00400000
#define DEP_TRACE           0x00800000
#define UTIL_TRACE          0x01000000

TraceLevel=3
TraceModule=0x0000

## Options to control the applied modules. The input may be partially
## processed, or not a full analysis may me wanted. The specific
## formats are a choice of the main program using the library, as well
## as the responsability of calling only the required modules.
## Valid input/output formats are: plain, token, splitted, morfo, tagged, parsed
InputFormat=plain
OutputFormat=tagged
```

```
# consider each newline as a sentence end
AlwaysFlush=no

#### Tokenizer options
TokenizerFile="$FREELINGSHARE/es/tokenizer.dat"

#### Splitter options
SplitterFile="$FREELINGSHARE/es/splitter.dat"

#### Morfo options
SuffixAnalysis=yes
MultiwordsDetection=yes
NumbersDetection=yes
PunctuationDetection=yes
DatesDetection=yes
QuantitiesDetection=yes
DictionarySearch=yes
ProbabilityAssignment=yes
NERecognition=yes
DecimalPoint=","
ThousandPoint="."
LocutionsFile=$FREELINGSHARE/es/locucions.dat
QuantitiesFile=$FREELINGSHARE/es/quantities.dat
SuffixFile=$FREELINGSHARE/es/sufixos.dat
ProbabilityFile=$FREELINGSHARE/es/probabilitats.dat
DictionaryFile=$FREELINGSHARE/es/maco.db
NPDataFile=$FREELINGSHARE/es/np.dat
PunctuationFile=$FREELINGSHARE/common/punct.dat
ProbabilityThreshold=0.001

## NEC options
NEClassification=no
NECFilePrefix=$FREELINGSHARE/es/nec/nec

## Sense annotation options (none,all,mfs)
SenseAnnotation=none
SenseFile=$FREELINGSHARE/es/senses16.db
DuplicateAnalysis=false

#### Tagger options
Tagger=hmm
TaggerHMMFile=$FREELINGSHARE/es/tagger.dat
TaggerRelaxFile=$FREELINGSHARE/es/constr_gram.dat
TaggerRelaxMaxIter=500
TaggerRelaxScaleFactor=670.0
TaggerRelaxEpsilon=0.001
TaggerRetokenize=yes
```

```
TaggerForceSelect=retok

#### Parser options
GrammarFile=$FREELINGSHARE/es/grammar-dep.dat

#### Dependence Parser options
DepRulesFile=$FREELINGSHARE/es/dep/dependences.dat
```

El fitxer de configuració està prou comentat per a poder comprendre'n l'estructura i funcionament. No obstant això, en les pròximes línies explicarem algunes modificacions i observarem els resultats obtinguts.

Suposem que tenim un fitxer de text que conté l'oració "Avui fa un matí molt maco però diuen que a la tarda plourà" (suposem que anomenem el fitxer `text.txt`). És important que aquest fitxer estigui en la codificació ISO-8859-1, ja que si no hi és, les paraules accentuades no s'analitzaran correctament. Analitzem aquest fitxer utilitzant el fitxer de configuració per defecte, fent:

```
analyzer -f es.cfg < text.txt
```

Obtenim la sortida següent per pantalla:

```
Avui avui RG 1
fa fer VMIP3S0 1
un un DI0MS0 0.958241
matí matí NCMS000 0.978261
molt molt RG 0.971264
maco maco AQ0MS0 1
però però CC 1
a a SPS00 1
la el DA0FS0 0.992986
tarda tarda NCFS000 0.968254
plourà ploure VMIF3S0 1
. . Fp 1
```

Si volem desar el resultat de l'anàlisi en un fitxer de sortida, podem redireccionar la sortida estàndard a un fitxer de la manera següent:

```
analyze -f es.cfg < text.txt > fitxersortida.txt
```

Com podem veure, la sortida obtinguda amb les opcions per defecte correspon a una anàlisi morfosintàctica desambiguada, ja que únicament ofereix

una categoria gramatical per a cada paraula. Com veiem, ofereix les categories mitjançant etiquetes *PAROLE*.

Adreça web recomanada

Podeu trobar una descripció detallada de les etiquetes *PAROLE* per a castellà a http://garraf.jpgevg.upc.es/freeling/doc/userman/parole-es.html.

Ara modificarem el fitxer de configuració perquè la sortida sigui una anàlisi morfològica sense desambiguar, és a dir, que ens ofereixi totes les etiquetes possibles per a cada paraula. Per a aconseguir això obrirem l'arxiu `es.cfg` i modificarem la línia següent:

```
OutputFormat=tagged
```

i posarem:

```
OutputFormat=morfo
```

Ara, si repetim l'anàlisi obtindrem la sortida següent:

```
Avui avui RG 1
fa fer VMIP3S0 1
un un DI0MS0 0.958241 un PI0MS000 0.0402809 1 Z 0.0014782
matí matí NCMS000 0.978261 matar VMIS1S0 0.0217391
molt molt RG 0.971264 molt DI0MS0 0.0229885 molt PI0MS000 0.00574713
maco maco AQ0MS0 1
però però CC 1
a a SPS00 1
la el DA0FS0 0.992986 ell PP3FSA00 0.00693255 la I 8.15594e-05
tarda tarda NCFS000 0.968254 tardar VMIP3S0 0.015873 tardar VMM02S0 0.015873
plourà ploure VMIF3S0 1
. . Fp 1
```

Ara la sortida està sense desambiguar i ens ofereix totes les possibles categories gramaticals al costat d'una probabilitat associada.

Finalment, podem posar el **OutputFormat=parsed** i obtindrem una anàlisi sintàctica superficial:

```
S_[
  sadv_[
    +(Avui avui RG -)
  ]
  grup-verb_[
    +verb_[
      +(fa fer VMIP3S0 -)
    ]
```

```
]
sn_[
  espec-ms_[
    +indef-ms_[
      +(un un DI0MS0 -)
    ]
  ]
  +grup-nom-ms_[
    +n-ms_[
      +(matí matí NCMS000 -)
    ]
    s-a-ms_[
      sadv_[
        +(molt molt RG -)
      ]
      +a-ms_[
        +(maco maco AQ0MS0 -)
      ]
    ]
  ]
]
coord_[
  +(però però CC -)
]
grup-sp_[
  +prep_[
    +(a a SPS00 -)
  ]
  sn_[
    espec-fs_[
      +j-fs_[
        +(la el DA0FS0 -)
      ]
    ]
    +grup-nom-fs_[
      +n-fs_[
        +(tarda tarda NCFS000 -)
      ]
    ]
  ]
]
grup-verb_[
  +verb_[
    +(plourà ploure VMIF3S0 -)
  ]
]
F-no-c_[
```

```
    +(. . Fp -)
  ]
]
```

3.3. Dades lingüístiques que proporciona el Freeling

Si volem observar les dades lingüístiques que proporciona el Freeling i no volem instal·lar tota l'aplicació, podem baixar l'arxiu `FreeLing-X.X.tar.gz` i descomprimir-lo. Si anem al subdirectori `/data` tindrem accés a les dades lingüístiques per a totes les llengües que tracta el Freeling. Si anem al subdirectori `/ca` tindrem accés a les dades corresponents al català. Podem destacar els fitxers següents:

- **dicc.src**: és el formari. Té l'estructura següent:

  ```
  casa casa NCFS000 casar VMIP3S0 casar VMM02S0 casar VMMP2S0
  casaca casaca NCFS000
  casada casada NCFS000 casar VMP00SF
  casades casada NCFP000 casar VMP00PF
  casador casador AQ0MS0
  casadora casador AQ0FS0
  casadores casador AQ0FP0
  ```

  És un fitxer de text separat per espais en blanc. Fixem-nos que en una mateixa línia tenim en primer lloc la forma i després totes les possibles anàlisis per a aquesta forma, amb el lema i l'etiqueta morfosintàctica. En la versió 2.1 aquest formari té un total de 640.899 entrades.

- **locucions.dat**: és un fitxer que conté locucions amb la seva informació morfosintàctica associada. El fitxer té l'estructura següent:

  ```
  a_bord a_bord RG I
  a_bots_i_barrals a_bots_i_barrals RG I
  a_brida_abatuda a_brida_abatuda RG I
  a_cabassos a_cabassos RG I
  ```

  En la versió 2.1 aquest fitxer conté un total de 1.699 entrades.

- **probabilitats.dat**: conté les probabilitats per a les diferents formes i etiquetes. Té l'estructura següent:

  ```
  casa NC-VMI-VMM NC 7 VMI 0 VMM 0
  ```

  Indica les possibles etiquetes per a una paraula i amb quina freqüència ha sortit amb cada etiqueta dins del corpus d'entrenament. Fixem-nos que *casa* ha sortit 7 vegades al corpus d'entrenament i sempre com a substantiu femení.

- **senses.src**: és el fitxer corresponent al *wordnet*. El fitxer està dividit en dues parts. La primera part té l'estructura següent:

  ```
  S:00005515:A absolut complet perfecte total
  ```

  i indica que el sentit 00005515, que correspon a un adjectiu, es pot expressar amb les paraules *absolut*, *complet*, *perfecte* i *total*.

  La segona part té l'estructura següent:

  ```
  W:absolut:A 00005515 00489625 00523702 01844500 00684153 02095346 00490201
  ```

  i indica que la paraula *absolut*, que és un adjectiu, pot tenir els significats indicats.

També, en el subdirectori common, on hi ha una sèrie de dades lingüístiques comunes. D'aquí es pot destacar el WordNet, en el fitxer `wnX.X.src`, en què `X.X` indica la versió de WordNet.

4. Apertium

4.1. Què és l'Apertium

L'Apertium (Armentano-Oller i altres, 2007; Zubizarreta i altres, 2009; Forcada, 2009) no és només un sistema de traducció automàtica de programari lliure que es distribueix sota la llicència GNU General Public License (GPL).

L'Apertium és més que un sistema de traducció automàtica. És una plataforma a partir de la qual es poden desenvolupar sistemes de traducció automàtica per a nous parells de llengües, o millorar i ampliar els parells de llengües existents.

És a dir, l'Apertium a més de contenir tot el que cal per a usar el sistema de traducció automàtica, també es distribueixen totes les eines necessàries per a crear nous sistemes. Així, doncs, la plataforma Apertium proporciona:

- Un motor de traducció independent dels parells de llengües.
- Eines per a gestionar les dades lingüístiques necessàries per a construir un sistema de traducció automàtica per a un parell de llengües determinat.
- Dades lingüístiques per a diversos parells de llengües.

L'Apertium fa servir una metodologia de traducció automàtica que s'anomena de transferència superficial. Aquesta metodologia fa una anàlisi sintàctica de l'oració en la llengua de partida i la transfereix a la llengua d'arribada per a poder generar l'oració d'arribada. Aquesta anàlisi sintàctica no és profunda, sinó superficial. Això vol dir que l'arbre sintàctic que es genera no és complet i poden faltar relacions entre alguns constituents. Aquesta metodologia de traducció automàtica està inicialment concebuda per a parells de llengües emparentades (com poden ser català-castellà, català-francès, castellà-portuguès, etc.). No obstant això, aquesta metodologia ha estat expandida per a poder tractar parells de llengües més divergents (català-anglès, per exemple).

En l'Apertium, el processament del text d'entrada es duu a terme en diverses etapes:

1) Desformatació: es tracten diversos formats d'entrada (text, RTF, HTML, etc.) i se separa el text per traduir de la informació de format.

2) Anàlisi morfològica: a cada paraula del text per traduir s'assignen totes les possibles informacions morfològiques.

3) Desambiguació categorial: es desambigua la informació morfològica assignada en l'etapa anterior, de manera que a cada paraula només li quedi una lectura possible.

4) Anàlisi sintàctica superficial: es construeix un arbre d'anàlisi superficial de la frase de partida.

5) Transferència sintàctica superficial: es transfereix l'arbre d'anàlisi superficial de la frase de partida a un arbre d'anàlisi superficial equivalent a la llengua d'arribada.

6) Transferència lèxica: es busca la traducció en la llengua d'arribada de les paraules de la frase de partida; de totes les traduccions possibles serà necessari escollir la més adequada.

7) Generació morfològica: es generen les formes adequades de les paraules traduïdes (en plural o singular, el temps i la persona per als verbs, etc.).

8) Reformatació: es recupera la informació de format per a recuperar el format del document original.

L'Apertium utilitza transductors d'estats finits per a les operacions de processament lèxic (anàlisi i generació morfològica, transferència lèxica), models ocults de Markov per a la desambiguació categorial i *chunking* multietapa basat en estats finits per a la transferència superficial. El format de les dades lingüístiques és senzill i està basat en XML. Aquestes dades es poden compilar amb les eines que proporciona el sistema al format intern, de manera que es poden aconseguir grans velocitats de processament.

4.2. Instal·lació d'Apertium des de repositoris Debian

En les distribucions de Linux basades en Debian (Ubuntu, per exemple) podem trobar l'Apertium en els repositoris i instal·lar-lo de manera molt senzilla. Per a verificar si Apertium es troba en els repositoris hem de fer:

```
apt-cache search apertium
apertium - Shallow-transfer machine translation engine
apertium-dbus - A D-Bus service for the Apertium machine translation system
apertium-en-ca - Apertium linguistic data to translate between English and Catalan
apertium-en-es - Apertium linguistic data to translate between English and Spanish
apertium-eo-ca - Apertium linguistic data to translate between Esperanto and Catalan
apertium-eo-es - Apertium linguistic data to translate between Esperanto and Spanish
apertium-es-gl - Apertium linguistic data to translate between Spanish and Galician
apertium-es-pt - Apertium linguistic data to translate between Spanish and Portuguese
apertium-es-ro - Apertium linguistic data to translate between Spanish and Romanian
```

```
apertium-eu-es - Apertium linguistic data to translate between Basque and Spanish
apertium-fr-ca - Apertium linguistic data to translate between French and Catalan
apertium-fr-es - Apertium linguistic data to translate between French and Spanish
apertium-oc-ca - Apertium linguistic data to translate between Occitan and Catalan
apertium-oc-es - Apertium linguistic data to translate between Occitan and Spanish
apertium-pt-ca - Apertium linguistic data to translate between Portuguese and Catalan
apertium-pt-gl - Apertium linguistic data to translate between Portuguese and Galician
apertium-tolk - Graphical user interface for Apertium
libapertium3-3.1-0 - Shared library for Apertium
libapertium3-3.1-0-dev - Development library for Apertium
liblttoolbox3-3.1-0 - Shared library for the lttoolbox
liblttoolbox3-3.1-0-dev - Development library for lttoolbox
lttoolbox - Apertium lexical processing modules and tools
apertium-es-ca - Apertium linguistic data to translate between Spanish and Catalan
```

Observem que apareixen tots els programes i dades lingüístiques relacionades amb aquesta aplicació. Per a instal·lar l'Apertium simplement hem de fer:

```
sudo apt-get install apertium
```

Ara podem instal·lar les dades lingüístiques que vulguem fent:

```
sudo apt-get install apertium-es-ca i les que vulguem
```

Aquesta manera d'instal·lar Apertium és la més recomanable. L'únic inconvenient és que no tenim un control total sobre la versió que instal·lem i ens hem de conformar amb la versió que hi hagi al repositori.

4.3. Instal·lació de l'Apertium a partir del codi font

4.3.1. Obtenció de l'Apertium

A la pàgina web d'Apertium podem trobar informació detallada sobre el sistema, els parells de llengües disponibles, una demostració en línia, etc. El projecte és a Source Forge. Des d'aquesta pàgina web podrem baixar el motor de traducció, les dades lingüístiques dels diferents parells de llengües i altres utilitats relacionades amb l'Apertium.

Adreces web recomanades

La pàgina web de l'Apertium és http://xixona.dlsi.ua.es/apertium-www/.
També podeu visitar Source Forge: http://sourceforge.net/projects/apertium/

4.3.2. Instal·lació de l'Apertium

A continuació presentem tots els passos necessaris per a instal·lar el motor de traducció automàtica Apertium i les dades lingüístiques corresponents a un

parell de llengües. Si es volen instal·lar les dades corresponents a més parells de llengües només cal repetir els passos indicats per a instal·lar un dels parells.

La instal·lació pot dependre de la distribució que feu servir i dels prerequisits que ja tingueu instal·lats. En els passos següents indiquem com s'instal·len tots els prerequisits. Es possible que alguns ja els tingueu instal·lats en el vostre sistema.

Notes sobre la instal·lació

Sovint es produeixen errors d'instal·lació amb les últimes versions publicades. És aconsellable intentar instal·lar-ne una de les últimes, però no realment l'última. Relacionat amb això mateix, de vegades també passa que els últims diccionaris publicats no funcionen correctament amb l'última versió de l'Apertium. Per tot això, és millor escollir la penúltima versió.

1) Instal·lació dels prerequisits

a) cpp:

```
sudo apt-get install cpp
```

b) build-essential:

```
sudo apt-get install build-essential
```

c) pkg-config:

```
sudo apt-get install pkg-config
```

d) libxml2:

```
sudo apt-get install libxml2
```

e) libxml2-dev:

```
sudo apt-get install libxml2-dev
```

f) libxml2-utils:

```
sudo apt-get install libxml2-utils
```

g) xsltproc:

```
sudo  apt-get install xsltproc
```

h) flex:

```
sudo  apt-get install flex
```

i) libpcre3-dev:

```
sudo apt-get install libpcre3-dev
```

2) Instal·lació de lttoolbox. Baixem l'última versió del web de descàrrega de l'Apertium. Una vegada baixada aconsellem copiar-la en el directori /opt. Hem de seguir els passos següents:

```
sudo cp lttoolbox...tar.gz /opt
cd /opt
sudo gunzip lttoolbox...tar.gz
sudo tar -xvf lttoolbox...tar
cd lttoolbox...
sudo ./configure
sudo make
sudo make install
```

3) Instal·lació del motor de traducció. Baixem l'última versió del web de descàrrega de l'Apertium i fem:

```
sudo cp apertium...tar.gc /opt
cd /opt
sudo gunzip apertium...tar.gz
sudo tar -xvf apertium...tar
cd apertium...
sudo ./configure
sudo make
sudo make install
```

4) Instal·lació de les dades lingüístiques per a un parell de llengües. En aquest exemple instal·larem les dades corresponents al parell castellà-català (es-ca). Si volem instal·lar més parells de llengües, simplement s'han de repetir els passos per als parells que es vulguin introduir. Baixem l'última versió del web de descàrrega de l'Apertium i fem:

```
sudo cp apertium-es-ca...tar.gz /opt
cd /opt
sudo gunzip apertium-es-ca...tar.gz
sudo tar -xvf apertium-es-ca...tar
cd apertium-es-ca...
sudo ./configure
sudo make
sudo make install
```

Problemes d'instal·lació en Ubuntu

En l'Ubuntu o el Kubuntu hi pot haver problemes quan intentem instal·lar els diccionaris. Si ens dóna un error, "error while loading shared libraries liblttoolbox3-3.0.0.so.0", haurem de baixar els paquets:

http://packages.ubuntu.com/hardy/libs/liblttoolbox3-3.0-0
http://packages.ubuntu.com/hardy/libdevel/liblttoolbox3-3.0-0-dev

Són paquets `.deb` que es poden instal·lar fent:

```
dpkg -i filename.deb
```

4.4. Execució de l'Apertium

Per a traduir amb l'Apertium podem escriure "apertium" en un terminal sense cap argument i ens apareixerà l'ajuda:

```
apertium
USAGE: apertium [-d datadir] [-f format] [-u] <translation> [in [out]]
 -d datadir       directory of linguistic data
 -f format        one of: txt (default), html, rtf, odt, docx, wxml, xlsx, pptx
```

```
-a               display ambiguity
-u               don't display marks '*' for unknown words
-m memory.tmx    use a translation memory to recycle translations
-o translation   translation direction using the translation memory,
                 by default 'translation' is used instead
translation      typically, LANG1-LANG2, but see modes.xml in language data
in               input file (stdin by default)
out              output file (stdout by default)
```

Si tenim un fitxer de text que s'anomena hola.txt i que conté "Hola bon dia. Avui fa sol.", per a traduir-lo amb l'Apertium simplement haurem de fer:

```
apertium ca-es hola.txt
Hola buenos días. Hoy hace solo.
```

Fixem-nos que la traducció apareix per pantalla. Si volem que la traducció es desi en un arxiu simplement hem de fer:

```
apertium ca-es hola.txt traduccio.txt
```

En els exemples que he posat el programa s'havia instal·lat mitjançant el repositori i les dades lingüístiques s'havien posat en la ubicació estàndard (/usr/share/apertium), i per aquest motiu no hem hagut d'indicar la ruta a les dades lingüístiques. Si instal·leu les dades en una altra ubicació caldrà indicar la ruta sencera. Les rutes solen ser una mica complexes (per exemple, /opt/apertium-es-ca-1.0.5). Per a facilitar l'execució del programa podem crear un enllaç simbòlic a aquesta ruta amb una ruta molt més curta. Si en el terminal escrivim:

```
sudo ln -s /opt/apertium-es-ca-1.0.5 /es-ca
```

Creem un enllaç en /es-ca a la ruta completa i podem executar l'Apertium fent:

```
apertium-translator /es-ca arxiuoriginal.txt arxiutraduit.txt
```

4.5. Ampliació dels diccionaris

Una de les característiques més interessants de l'Apertium és que és un sistema de traducció automàtica de programari lliure, i això implica que el codi del programa, les regles i els diccionaris són accessibles i modificables per qualsevol usuari amb prou coneixements tècnics. Modificar el programa i les regles

és una tasca realment complicada. L'ampliació de diccionaris sí que és relativament senzilla i l'explicarem en aquest subapartat a partir d'un exemple assequible.

Volem traduir la frase en català "Tinc un diccionari terminològic de farmàcia i farmacologia". Si proveu de traduir-la veureu que l'Apertium no té les entrades corresponents a *terminològic* i a *farmacologia* i que el resultat de la traducció serà alguna cosa de l'estil: "Tengo un diccionario *terminològic de farmacia y *farmacologia", en què ens indica amb l'asterisc les paraules desconegudes.

Per a entrar les paraules desconegudes haurem d'actuar sobre tres diccionaris (que es troben en el directori on hem instal·lat els diccionaris):

- Diccionari monolingüe català (`apertium-es-ca.ca.dix`)
- Diccionari monolingüe castellà (`apertium-es-ca.es.dix`)
- Diccionari bilingüe castellà-català (`apertium-es-ca.es-ca.dix`)

Aquests diccionaris estan en un format XML i es poden editar amb qualsevol editor de textos. Per a començar, introduirem "farmacología - farmacologías". Editarem l'arxiu `apertium-es-ca.ca.dix`. Segons aquest arxiu cal introduir la paraula en català i informació sobre la seva categoria morfològica i la flexió. La flexió es dóna mitjançant un exemple establert. Podem buscar una paraula que tingui la mateixa flexió que farmacologia (considerarem que té de plural *farmacologies*), com per exemple *psicologia*, i veiem que és de la manera següent:

```
<e lm="psicologia">
    <i>psicologi</i>
    <par n="abell/a__n"/>
</e>
```

i escrivim una nova entrada aprofitant aquesta que sigui:

```
<e lm="farmacologia">
    <i>farmacologi</i>
    <par n="abell/a__n"/>
</e>
```

Ara hem de fer el mateix per a l'entrada en castellà, editem el diccionari `apertium-es-ca.es.dix` prenent com a exemple *psicología*:

```
<e lm="psicología">
    <i>psicología</i>
    <par n="abeja__n"/>
</e>
```

i escrivim una nova entrada aprofitant aquesta que sigui:

```
<e lm="farmacología">
    <i>farmacología</i>
    <par n="abeja__n"/>
</e>
```

Podem escriure aquesta entrada on vulguem, però per a controlar millor els canvis ho farem gairebé al final de l'arxiu, abans de la secció `</section> <section id="final"type=ïnconditional»`.

Ara ens queda modificar el diccionari bilingüe `apertium-es-ca.es-ca.dix`. Buscarem l'entrada corresponent a "psicología - psicologia":

```
<e>
  <p>
    <l>psicología<s n="n"/></l>
    <r>psicologia<s n="n"/></r>
  </p>
</e>
```

i gairebé al final de l'arxiu, abans de la secció `</section> <!- <section id="final"type="standard»` crearem l'entrada corresponent a "farmacología - farmacologia".

```
<e>
  <p>
    <l>farmacología<s n="n"/></l>
    <r>farmacologia<s n="n"/></r>
  </p>
</e>
```

Ara hem de fer els mateixos passos per a l'adjectiu "terminològic - terminológico". Repetiu-los de la mateixa manera utilitzant com a exemple en català *analògic*, i en castellà *analógico*. Una vegada modificats els diccionaris s'ha de tornar a fer en el directori que conté els diccionaris:

```
sudo make
sudo make install
```

4.6. Fitxers de dades lingüístiques proporcionats per Apertium

En aquest subapartat analitzarem quins fitxers de dades lingüístiques proporciona l'Apertium i com es poden manipular per a fer-los servir en les nostres aplicacions pròpies. Tot i que hàgim instal·lat l'Apertium al nostre ordinador tornarem a baixar les dades lingüístiques per al parell de llengües que vulguem (en el nostre cas baixarem el fitxer corresponent a anglès-català) des de http://sf.net/projects/apertium/. Un cop baixades descomprimirem el arxiu. Ens fixarem en dos arxius:

- **apertium-en-ca.ca.dix**: és el diccionari a partir del qual es poden obtenir totes les formes flexionades per al català.
- **apertium-en-ca.en-ca.dix**: és el diccionari bilingüe anglès-català.

En els propers subapartats veurem amb més detall aquests dos arxius.

4.6.1. L'arxiu apertium-en-ca.ca.dix

Aquest diccionari permet generar totes les formes per al català. L'arxiu té diversos apartats diferenciats:

- **alphabet**: aquí es defineix l'alfabet emprat, és a dir, totes les lletres possibles per a la llengua en qüestió.
- **sdefs**: definició de les etiquetes per a les categories gramaticals i les subcategoritzacions.
- **pardefs**: definició dels paradigmes.
- **section**: és el diccionari pròpiament dit.

El primer programa que farem (`programa-10-4.py`) simplement traurà una llista de tots els lemes presents al diccionari.

```
import sys, codecs
import xml.sax
import xml.sax.handler

class ApertiumHandler(xml.sax.handler.ContentHandler):
    """
    A handler to deal with Apertium dictionary format
    """
    def __init__(self):
        self.inSection=False
        self.inEntrada=False
        self.entrada=""
        self.diccionari=[]
    def startElement(self,name,attrs):
```

```
        if name=="section":
            self.inSection=True
        elif name=="e":
            self.inEntrada=True
            if attrs.has_key(l·lm"):
                self.entrada=attrs[l·lm"]
                self.diccionari.append(self.entrada)

    def endElement(self,name):
        if name=="section":
            self.inSection=False
        elif name=="e":
            self.inEntrada=False
    def returnDictionary(self):
        self.sortedDictionary=sorted(self.diccionari)
        return self.sortedDictionary

sys.stdout=codecs.lookup('utf-8')[-1](sys.stdout)

parser=xml.sax.make_parser()
handler=ApertiumHandler()
parser.setContentHandler(handler)
parser.parse("apertium-en-ca.ca.dix")

diccionari=handler.returnDictionary()

for entrada in diccionari:
    print entrada
```

Aquest programa emmagatzem l'atribut *lm* (lema) de les entrades en una llista. Amb el mètode **returnDictionary** retornem una versió ordenada alfabèticament d'aquesta llista. Ara volem ampliar el programa perquè faci una llista de totes les formes. Caldrà llegir la informació relativa a la flexió per a cada un dels paradigmes i aplicar-la a la pseudoarrel que proposa a cada entrada. El programa ampliat el podeu trobar a `programa-10-5.py`:

```
import sys, codecs
import xml.sax
import xml.sax.handler

class ApertiumHandler(xml.sax.handler.ContentHandler):
    """
    A handler to deal with Apertium dictionary format
    """
    def __init__(self):
        self.inSection=False
```

```
        self.inEntrada=False
        self.inPardef=False
        self.inParadigma=False
        self.inI=False
        self.inPar=False
        self.inL=False
        self.inR=False
        self.entrada=""
        self.nomPara=""
        self.pseudoarrel=""
        self.paradigma=""
        self.terminacioLema=""
        self.terminacioForma=""
        self.diccionari=[]
        self.hashDict={}
        self.paradigmes={}
    def startElement(self,name,attrs):
        if name=="section":
            self.inSection=True
        elif name=="e":
            self.inEntrada=True
            if attrs.has_key(l·lm"):
                self.entrada=attrs[l·lm"]
                e=self.entrada+":"+self.entrada
                self.hashDict[e]=1
        elif name=="pardef":
            self.inPardef=True
            if attrs.has_key("n"):
                self.nomPara=attrs["n"]
        elif name=="p":
            self.inParadigma=True
        elif name==l·l":
            self.inL=True

        elif name=="r":
            self.inR=True

        elif name==ï":
            self.inI=True

        elif name=="par":
            self.inPar=True
            if attrs.has_key("n"):
                self.paradigma=attrs["n"]

    def characters(self,data):
        if self.inPardef:
```

```
            if self.inEntrada:
                if self. inL:
                    self.terminacioForma=data
        if self.inI:
            self.pseudoarrel=data

    def endElement(self,name):
        if name=="section":
            self.inSection=False
        elif (name=="e" and self.inPardef==False):
            self.inEntrada=False
            if (self.entrada.find(" ") == -1):
                for terminacio in self.paradigmes[self.paradigma].split(":"):
                    e=self.pseudoarrel+terminacio+":"+self.entrada
                    self.hashDict[e]=1
        elif name=="pardef":
            self.inPardef=False
            self.terminacioLema=""
            self.terminacioForma=""
        elif name==l·l":
            self.inL=False
        elif name=="r":
            self.inR=False
        elif name=="p":
            self.inParadigma=False
            self.afegeixTerminacioForma(self.nomPara,self.terminacioForma)
        elif name==ï":
            self.inI=False

        elif name=="par":
            self.inPar=False

    def afegeixTerminacioForma(self, nP, tF):
        if self.paradigmes.has_key(nP):
            self.paradigmes[nP]+=":"+tF
        else:
            self.paradigmes[nP]=tF

    def returnDictionary(self):

        self.diccionari=self.hashDict.keys()
        self.sortedDictionary=sorted(self.diccionari)
        return self.sortedDictionary

sys.stdout=codecs.lookup('utf-8')[-1](sys.stdout)

parser=xml.sax.make_parser()
```

```
handler=ApertiumHandler()
parser.setContentHandler(handler)
parser.parse("apertium-en-ca.ca.dix")

diccionari=handler.returnDictionary()

for entrada in diccionari:
    print entrada
```

En la sortida d'aquest programa tindrem la forma i el lema corresponent separats per dos punts (":").

Exercici

Modifiqueu aquest programa per a fer que en la sortida tinguem també la categoria gramatical.

4.6.2. L'arxiu apertium-en-ca.en-ca.dix

En aquest fitxer podem trobar el diccionari de transferència, és a dir, el diccionari bilingüe anglès-català. Farem un programa que tracti la secció

```
<section  id="main" type="standard»
```

d'aquest fitxer i que retorni un diccionari anglès-català en format de text separat per tabuladors. Fixem-nos que en aquesta secció les entrades tenen l'estructura següent:

```
<e><p><l>formative<s n="adj"/></l><r>formatiu<s n="adj"/></r></p></e>
```

La llista del programa (`programa-10-6.py`) és la següent:

```
import sys, codecs
import xml.sax
import xml.sax.handler

class ApertiumHandler(xml.sax.handler.ContentHandler):
    """
    A handler to deal with Apertium bilingual dictionary format
    """
    def __init__(self):
        self.inMainSection=False
        self.inEntrada=False
        self.inP=False
```

```
        self.inL=False
        self.inR=False
        self.eng=""
        self.cat=""
        self.diccionari={}
    def startElement(self,name,attrs):
        if name=="section":
            if (attrs.has_key(ïd") and attrs.has_key("type")):
                if (attrs[ïd"]=="main" and attrs["type"]=="standard"):
                    self.inMainSection=True
        elif name=="e":
            self.inEntrada=True
        elif name=="p":
            self.inP=True
        elif name==l·l":
            self.inL=True
        elif name=="r":
            self.inR=True
    def characters(self,data):
        if (self.inMainSection and self.inEntrada and self.inP and self.inL):
            self.eng+=data
        elif (self.inMainSection and self.inEntrada and self.inP and self.inR):
            self.cat+=data
    def endElement(self,name):
        if name=="section":
            self.inMainSection=False
        elif name=="e":
            self.inEntrada=False
        elif name=="p":
            self.inP=False
            if (self.inMainSection and self.inEntrada):
                print self.eng, self.cat
                self.eng=""
                self.cat=""
        elif name==l·l":
            self.inL=False
        elif name=="r":
            self.inR=False
        elif name=="b":
            self.eng+=" "
            self.cat+=" "

    def returnDictionary(self):
        self.sortedDictionary=sorted(self.diccionari)
        return self.sortedDictionary
```

```
sys.stdout=codecs.lookup('utf-8')[-1](sys.stdout)

parser=xml.sax.make_parser()
handler=ApertiumHandler()
parser.setContentHandler(handler)
parser.parse("apertium-en-ca.en-ca.dix")

diccionari=handler.returnDictionary()

for entrada in diccionari:
    print entrada
```

Exercici

Amplieu el diccionari perquè us retorni també la categoria gramatical.

Resum

En aquest mòdul hem presentat un seguit d'eines de programari lliure que porten a terme algun tipus de processament per al català. La utilitat d'aquestes eines és doble: per una banda el processament que fan i per l'altra les dades lingüístiques que proporcionen. Si fem servir les dades lingüístiques en els nostres programes haurem de respectar les condicions de llicència particular de cada eina i dades. Les diferents utilitats que hem presentat tenen les llicències:

- El Corrector: GPL
- Dacco: LGPL (l'aplicació de consulta qdacco té llicència GPL)
- Freeling: GPL
- Apertium: GPL

Podeu trobar tots els detalls sobre la llicència GPL (*General Public License*, Llicència Pública General) a http://www.gnu.org/licenses/licenses.ca.html. La idea bàsica d'aquesta llicència és que els programes o recursos es poden fer servir de manera lliure, compartir, modificar, etc., però els programes o recursos que es derivin hauran de tenir la mateixa llicència o una de compatible. En canvi, la LGPL (*Lesser General Public License*, Llicència Pública General Reduïda) no obliga al darrer punt.

Bibliografia

Armentano-Oller, C.; Corbi-Bellot, A.; Forcada, M.; Ginesti-Rosell, M.; Belda, M.; Ortiz-Rojas, S.; Pérez-Ortiz, J.; Ramirez-Sánchez, G. i Sánchez-Martinez, F. (2007). «Apertium, una plataforma de código abierto para el desarrollo de sistemas de traducción automática». A: «Proceedings of the FLOSS International Conference», (pàgs. 5–20). Servicio de publicaciones de la Universidad de Cadiz.

Carreras, X.; Chao, I.; Padró, L. i Padró, M. (2004). «Freeling: An open-source suite of language analyzers». A: «Proceedings of the 4th LREC», volum 4.

Forcada, M. (2009). «Apertium: traducció automàtica de codi obert per a les llengües romàniques». A: *Linguamática*, volum 1(1): pàg. 13.

Martí Quixal, T. B. F. B. J. R. B. J. D. B. G. G. M. i Valentín, O. (2008). «User-Centred Design of Error Correction Tools». A: «Proceedings of the Sixth International Language Resources and Evaluation (LREC'08)», (ed.) **Nicoletta Calzolari (Conference Chair), K. C. B. M. J. M. J. O. S. P. D. T.** Marrakech, Morocco: European Language Resources Association (ELRA). ISBN 2-9517408-4-0. Http://www.lrec-conf.org/proceedings/lrec2008/.

Zubizarreta, F.; Mikel, L.; Tyers, F. i Ramírez-Sánchez, G. (2009). «The Apertium machine translation platform: five years on».