# UAM Text Tools adaptation to Spanish

Gabriel Rodríguez Rodríguez

1. Learn and try examples with UAM text tools http://utt.amu.edu.pl/
2. Search for a spanish morphologic dictionary
   - We found it in the FreeLing project http://nlp.lsi.upc.edu/freeling/
   - You can download the tar.gz with the files and search the spanish morphologic dictionary in FreeLing-2.2.2/data/es/dicc.src
3. Convert our dictionary to the UTT dictionary format, needed for use it in the utt tools.
   - I did a python script for analyze the dictionary and create a new file in the correct format
   - Before, we have to do a table conversion because there aren't the same kind of verbs, prepositions, adverbs... in spanish than in polish
   - In the spanish dicctionary, when we have more than one kind of word for the same word, the are in the same line. For our new dictionary, same words will be splitted in different words, with only one word per line
   - This will be the code of the python script. You have to specified the input file and the name of the output file:

```
import string

def codeNoum(code):
        p1="N/"

        #Analyze Gender
        if code[2]=='M':p2='Gp'
        elif code[2]=='F':p2='Gf'
        elif code[2]=='C':p2='Gn'

        #Analyze Number
        p3='N'+code[3].lower()

        return p1+p2+p3

def codeVerb(code):
        p1="V/"

        #Analyze Type TY
        p2='TY'+code[1].lower()

        #Analyze Mode M
        if   code[2]=='I':p3='Md'
        elif code[2]=='S':p3='Ms'
        elif code[2]=='M':p3='Mi'
        elif code[2]=='N':p3='Mb'
        elif code[2]=='G':p3='Mg'
        elif code[2]=='P':p3='Mp'

        #Analyze Time T
        if   code[3]=='P':p4='Tr'
        elif code[3]=='I':p4='Ti'
        elif code[3]=='F':p4='Tf'
        elif code[3]=='S':p4='Tp'
        elif code[3]=='C':p4='Tc'
        elif code[3]=='0':p4=''

        #Analyze Person
        p5='P'+code[4]

        #Analyze Number
        p6='N'+code[5].lower()

        #Analyze Gender
        if code[6]=='M':p7='Gp'
        if code[6]=='F':p7='Gf'
        else: p7=''

        return p1+p2+p3+p4+p5+p6+p7


def codeAdj(code):
        p1="ADJ/"


        #Analyze Type T
        if   code[1]=='Q':p2='Tq'
        elif code[1]=='O':p2='To'
        else: p3=''

        #Analyze Degree D
        if code[2]=='C':  p3='Dc'
        elif code[2]=='A':p3='Da'
        elif code[2]=='D':p3='Dd'
        elif code[2]=='S':p3='Ds'
        else: p3=''

        #Analyze Gender G
        if   code[3]=='M':p4='Gp'
        elif code[3]=='F':p4='Gf'
        elif code[3]=='C':p4='Gc'
        else: p4=''

        #Analyze Number N
        if   code[4]=='S':p5='Ns'
        elif code[4]=='P':p5='Np'
        elif code[4]=='N':p5='Nn'
        else: p5=''

        #Analyze Function F
        if code[5]=='P':p6='Fp'
        else: p6=''

        return p1+p2+p3+p4+p5+p6

def codeAdv(code):
        p1="ADV/"


        #Analyze Type T
        if   code[1]=='G':p2='Tg'
        elif code[1]=='N':p2='Tn'
        else: p2=''

        return p1+p2


def codeDet(code):
        p1="D/"


        #Analyze Type T
        if   code[1]=='D':p2='Td'
        elif code[1]=='P':p2='Tp'
        elif code[1]=='T':p2='Tt'
        elif code[1]=='E':p2='Te'
        elif code[1]=='I':p2='Ti'
        elif code[1]=='A':p2='Ta'
        else: p2=''

        #Analyze Person
        p3='P'+code[2]

        #Analyze Gender G
        if   code[3]=='M':p4='Gp'
        elif code[3]=='F':p4='Gf'
        elif code[3]=='C':p4='Gc'
        elif code[3]=='N':p4='Gn'
        else: p4=''

        #Analyze Number N
        if   code[4]=='S':p5='Ns'
        elif code[4]=='P':p5='Np'
        elif code[4]=='N':p5='Nn'
        else: p5=''

        #Analyze Owner
        if   code[5]=='S':p6='Os'
        elif code[5]=='P':p6='Op'
        else: p6=''

        return p1+p2+p3+p4+p5+p6



def codePro(code):
        p1="PRO/"

        #Analyze Type T
        if   code[1]=='P':p2='Tp'
        elif code[1]=='D':p2='Td'
        elif code[1]=='X':p2='Tx'
        elif code[1]=='I':p2='Ti'
        elif code[1]=='T':p2='Tt'
        elif code[1]=='R':p2='Tr'
        elif code[1]=='E':p2='Te'
        else: p2=''

        #Analyze Person
        p3='P'+code[2]

        #Analyze Gender G
        if   code[3]=='M':p4='Gp'
        elif code[3]=='F':p4='Gf'
        elif code[3]=='C':p4='Gc'
        elif code[3]=='N':p4='Gn'
        else: p4=''

        #Analyze Number N
        if   code[4]=='S':p5='Ns'
        elif code[4]=='P':p5='Np'
        elif code[4]=='N':p5='Nn'
        else: p5=''

        #Analyze Case C
        if   code[5]=='N':p6='Cn'
        elif code[5]=='A':p6='Ca'
        elif code[5]=='D':p6='Cd'
        elif code[5]=='O':p6='Co'
        else: p6=''

        #Analyze Owner O
        if   code[6]=='S':p7='Os'
        elif code[6]=='P':p7='Op'
        else: p7=''

        #Analyze Polite POL
        if code[7]=='P':p8='POLp'
        else: p8=''

        return p1+p2+p3+p4+p5+p6+p7+p8


def codeConj(code):
        p1="CONJ/"

        #Analyze Type T
        if   code[1]=='C':p2='Tc'
        elif code[1]=='S':p2='Ts'
        else: p2=''

        return p1+p2


def codeInt(code):
        p1="I"
        return p1


def codeConj(code):
        p1="CONJ/"

        #Analyze Type T
        if   code[1]=='C':p2='Tc'
        elif code[1]=='S':p2='Ts'
        else: p2=''

        return p1+p2


def codePrep(code):
        p1="P/"

        #Analyze Type T
        p2='Tp'

        #Analyze Form F
        if   code[2]=='S':p3='Fs'
        elif code[2]=='F':p3='Ff'
        else: p3=''

        #Analyze Gender G
        if   code[3]=='M':p4='Gp'
        elif code[3]=='F':p4='Gf'
        else: p4=''

        #Analyze Number N
        if   code[4]=='S':p5='Ns'
        elif code[4]=='P':p5='Np'
        else: p5=''

        return p1+p2+p3+p4+p5


def  compareBase(word, base):
        index=0;
        while index < len(word):
                if (index<len(base)) and (word[index] == base[index]):
                        index=index+1
                        continue
                codeInt=len(word)-index
                code=str(codeInt)+base[index::]
                return code
        return 0


# MAIN

f = open ("nuevoDicc.txt", "w")

for line in file('dicc8.src'):

        splited = line.split()

        for block in range(len(splited)/2):

                code = splited[2*(block+1)]
                if code[0]=='N':
                        nuevo=codeNoum(code)
                elif code[0]=='V':
                        nuevo=codeVerb(code)
                elif code[0]=='A':
                        nuevo=codeAdj(code)
                elif code[0]=='R':
                        nuevo=codeAdv(code)
                elif code[0]=='D':
                        nuevo=codeDet(code)
                elif code[0]=='P':
                        nuevo=codePro(code)
                elif code[0]=='C':
                        nuevo=codeConj(code)
                elif code[0]=='I':
                        nuevo=codeInt(code)
                elif code[0]=='S':
                        nuevo=codePrep(code)

                f.write(splited[0]+";"+str(compareBase(splited[0],splited[2*block+1]))+","+nuevo+"\n")

  # line contiene la linea actual

f.close()
```

4. Change program tok (for tokenizer) adding spanish characters and removing polish characters. We also added “¡” and “¿” characters as punctuation characters. We will work always with latin1.

***tok.l***

```
%{
        #include <stdio.h>
        #include <locale.h>
        #include "tok_cmdline.h"

        int filepos=0;

        struct gengetopt_args_info args;

%}

%%

[a-zA-ZáÁéÉíÍóÓúÚüÜñÑ]{1,64}            {
                                                printf("%04d %02d W %s\n", filepos, yyleng, yytext);
                                                filepos+=yyleng;
                                                if(args.interactive_flag) fflush(stdout);
```

```
                                        }

[[:digit:]]{1,64}               {
                                                printf("%04d %02d N %s\n", filepos, yyleng, yytext);
                                                filepos+=yyleng;
                                                if(args.interactive_flag) fflush(stdout);
                                        }

[[:space:]\n]{1,64}                     {
                                                int i;
                                                printf("%04d %02d S ", filepos, yyleng);
                                                for(i=0; i<yyleng; ++i)
                                                        switch(yytext[i])
                                                        {
                                                                case ' ' : putchar('_'); break;
                                                                case '\t': printf("\\t"); break;
                                                                case '\n': printf("\\n"); break;
                                                                case '\r': printf("\\t"); break;
                                                                case '\f': printf("\\n"); break;
                                                        }
                                                putchar('\n');
                                                filepos+=yyleng;
                                                if(args.interactive_flag) fflush(stdout);
                                        }

[[:punct:]]                     {
                                                printf("%04d %02d P %c\n", filepos, yyleng, *yytext);
                                                filepos+=yyleng;
                                                if(args.interactive_flag) fflush(stdout);
                                        }

[¡¿]                                    {

                                                printf("%04d %02d P %c\n", filepos, yyleng, *yytext);

                                                filepos+=yyleng;

                                                if(args.interactive_flag) fflush(stdout);

                                        }


.                                       {
                                                printf("%04d %02d B \\x%02X\n", filepos, yyleng, (unsigned char)*yytext);
                                                filepos+=yyleng;
                                                if(args.interactive_flag) fflush(stdout);
                                        }

%%

int main(int argc, char** argv)
{
  if (cmdline_parser(argc, argv, &args) != 0) return 1;
  setlocale(LC_CTYPE,"");
  setlocale(LC_COLLATE,"");
  yylex();
  return 0;
}

int yywrap()
{
        return 1;
}
```

5. For use the “sen” program (for split text in sentences), we add spanish abreviations in

the sen.l file and tried to do that detect the “¿¡” characters like start sentences characters

ab1 (Abg|a.C|d.C|admÃ³n|prof|adm|Arq|Arz|atte|atto|av|avd|avda|Bibl|c|cap|Cap|Cdad|cent|cent|cÃ³d|Comte|coord|crec|cta|D|D.Âª|diag|dicc|depto|DÃ±a|doc|Dr|Dra|Dr.Âª|dto|edit|EE. UU|EE.UU|Excmo|Excma|F. C|F.C|Fdo|Gral|Hno|Hna|impto|incl|Ing|Inst|JJ. OO|JJ.OO|Lcdo|Lcda|Ldo|Lda|Ltd|mÃ¡x|mÃn|nÃºm|No|Ob|p|pg|pÃ¡g|pÃ¡rr|P. D|P.D|pdo|ej|pl|plza|ppal|prÃ³l|prov|pza|Rep|R.I.P|D.E.P|RR. HH|RR.HH|Rte|sig|Srta|tel|telÃ©f|tfno|trad|V|Vd|Vds|V. O|V.O|V. O. S|V.O.S)

6. For use our diccionary in lem program, we must to compile our dictionary, because it’s so big for use it directly like text .dic file. We will do it with “compdic” tool
    - First, we must recode our dicctionary from utf8 to latin1 with “recode u8..l1 file”
    - We also have to edit the compdic program adding new spanish characters that they are in our dictionary and remove polish characters. The new compdic file will be like this:

```
no_of_parts=0


while [ $# -gt 2 ]
do
  case $1
  in
    -p)
      no_of_parts=$2
      shift 2
    ;;

    *)
      echo "The arguments to use are"
      echo "-p: number of parts"
      shift 1
    ;;
  esac
done

if [ $# -lt 2 ]
then
    echo "Usage:"
    echo "        compdic [-p <parts>] <wordlist> <automaton>"
    echo "where"
    echo "    <wordlist> - file containig a list of words, one per line, iso-8859-2 encoded"
    echo "    <automaton> - a file to which the compiled automaton (cor/kor format) shoul be written"
    exit 0
fi
```

```
if [ $no_of_parts -eq 0 ]
then
    no_of_parts=$(( `cat $1 | wc -l` / 75000 + 1 ))
fi


echo number of parts: $no_of_parts
tempdir=`mktemp -d /tmp/compdic.XXXXXX`

alphabet=`tempfile -d $tempdir`
cat <<EOF > $alphabet

<eps> 0
a 1
A 2
á 3
Á 4
b 6
B 7
c 8
C 9
é 10
É 11
d 12
D 13
e 14
E 15
f 19
F 20
g 21
G 22
h 23
H 24
i 25
I 26
j 27
J 28
k 29
K 30
l 31
L 32
í 33
Í 34
m 35
M 36
n 37
N 38
ñ 39
Ñ 40
o 41
O 42
ó 44
Ó 45
p 46
P 47
q 48
Q 49
r 50
R 51
s 52
S 53
```

```
ú 54
Ú 55
t 56
T 57
u 58
U 59
v 61
V 62
w 63
W 64
x 65
X 66
y 67
Y 68
z 69
Z 70
0 75
1 76
2 77
3 78
4 79
5 80
6 81
7 82
8 83
9 84
_ 85
- 86
? 87
! 88
~ 89
; 90
, 91
/ 92
* 93
+ 94
. 95
ü 96
· 97
ê 98
à 99
ç 100
ö 101
ï 102
º 103
è 104
ë 105
ä 106
Å 107
EOF

no_of_lines=$(( (`cat $1 | wc -l` / $no_of_parts) + 1 ))
split -l $no_of_lines $1 $tempdir/part
automaton=$tempdir/output.fst

cat <<EOF | fstcompile --acceptor --isymbols=$alphabet > $automaton

EOF
n=0
for f in $tempdir/part.*
do
```

```
    temp1=`tempfile -d $tempdir`
    temp2=`tempfile -d $tempdir`
    temp3=`tempfile -d $tempdir`
    n=$(( $n + 1 ))
    echo processing part $n
    cat $f |\
    lst2fstext |\
    fstcompile --acceptor --isymbols=$alphabet |\
    fstrmepsilon |\
    fstdeterminize > $temp1
    fstminimize $temp1 $temp2

    fstunion $automaton $temp2 | fstrmepsilon | fstdeterminize > $temp3
    fstminimize $temp3 $automaton

done

cat $automaton | fsttopsort | fstprint --acceptor --isymbols=$alphabet > aaaa
cat aaaa | fsm2aut | aut2fsa > $2
rm -r $tempdir
```

7. We will compile our dictionary with “compdic dicname.dic dicname.bin”
8. We can use the “sen” comand with parameter -d dicname.bin for use the new dictionary compiled