Informe de medicacion

Ver programas en github: https://github.com/asqwerty666/medica

tl;dr

El formato de la base de datos de input ha de ser:

```
<ID_Subject>;<ID_visita>;<Texto libre>
```

y se ejecutan dos scripts consecutivos,

```
$ ./parse.pl medicamentos.db
$ ./apply_rules.pl -r med_rules_ed_0219.txt
```

Y los resultados quedan en el archivo *meds_matrix.csv*. 😉

Antes de empezar

Lo primero es que el formato de *input* ha de ser fijo. El archivo que tengamos hay que llevarlo a ese formato.

Ejemplo: tenemos,

```
[osotolongo@brick03 medicacion]$ head anamnesi.csv
;Interno;xlinea_id;Fecha_A_DN_EXN;Fecha_A_DN_EXN_Modificacion;Fecha_A_DN_EXN
_Cierre;Recetas_SIRE
1;19960001;1;1997-04-30;1997-04-30 00:00:00.000;1997-04-30 00:00:00.000;
2;19960003;165;1996-01-29;1996-01-29 00:00:00.000;1996-01-29
00:00:00.000;Boi-k Seguril Plurimen Aremis 50 1-0-0 Masdil Tensoprem
3;19960004;252;1996-01-01;1996-01-01 00:00:00.000;1996-01-01
00:00:00.000;Meleril 50 Distraneurine Cisordimol 0-0-5 Nerdipina 1-1-1
4;19960006;317;1996-01-17;1996-01-17 00:00:00.000;1996-01-17
00:00:00.000;Escazine 1-0-0 Remontal 1-1-1 Ciclofalina 3-3-0 Meleril 5-5-5
Aneurol si precisa Hidroferol 1 al m,s
5;19960008;334;1996-01-29;1996-01-29 00:00:00.000;1996-01-29 00:00:00.000;
6;19960009;341;1996-01-16;1996-01-16 00:00:00.000;1996-01-16
00:00:00.000;Eskazine 2mg cada 8 hores Largactil intramuscular en cas
d'agitaci¢
7;19960010;2;1996-02-26;1996-02-26 00:00:00.000;1996-02-26
00:00:00.000;Becozyme 1/8h
8;19960010;8692;1996-02-26;1996-02-26 00:00:00.000;1996-02-26 00:00:00.000;
9;19960011;10;1996-01-16;1996-01-16 00:00:00.000;1996-01-16 00:00:00.000;
```

Asi que hacemos,

```
[osotolongo@brick03 medicacion]$ awk -F";" '{print $2";"$4";"$7}'
anamnesi.csv | sed 's/-//;s/-//' > anamnesi.db
```

y nos quedamos con,

```
[osotolongo@brick03 medica]$ head anamnesi.db
Interno;Fecha_A_DN_EXN;Recetas_SIRE
19960001;19970430;
19960003;19960129;Boi-k Seguril Plurimen Aremis 50 1-0-0 Masdil Tensoprem
19960004;19960101;Meleril 50 Distraneurine Cisordimol 0-0-5 Nerdipina 1-1-1
19960006;19960117;Escazine 1-0-0 Remontal 1-1-1 Ciclofalina 3-3-0 Meleril
5-5-5 Aneurol si precisa Hidroferol 1 al m,s
19960008;19960129;
19960009;19960116;Eskazine 2mg cada 8 hores Largactil intramuscular en cas
d'agitaci¢
19960010;19960226;Becozyme 1/8h
19960010;19960226;
19960011;19960116;
```

Ojo aqui: El comando *awk* selecciona lsolo las columnas que necesitamos. Esto incluye el *ID*, la fecha de cada visita y el texto libre que se escribe en la receta. A las fechas le estoy quitando el caracter - y el texto libre lo dejo tal cual. De este ultimo campo hay mucha porqueria que quitar pero eso lo voy a hacer en el parser directamente.

---

o en el caso de,

```
[osotolongo@brick03 medicacion]$ head seguimientos.csv
Interno;FechaSeguiment;FechaSeguiment_Modificacion;FechaSeguiment_Cierre;Rec
eta_SIRE_SN
19960014;1998-12-17;12/17/1998;12/17/1998;
19960044;2005-03-14;3/14/2005;3/14/2005;igual: tegretol:1/2-0-0
19960068;1999-07-13;7/13/1999;7/13/1999;
19960171;2010-03-31;3/31/2010;3/31/2010;Reminyl 24 mgr LR 1-0-0, Axura 20
mgr, Omeprazol 20 mgr 1-0-0, Pazital, Nerdipina 1-0-1, Prevencor 0-1-0,
Sumial 10 mgr, Sedotime 0-0-1 , Plavix 0-1-0.
19960171;2011-03-29;3/29/2011;3/29/2011;Dormicum , Esertia 10 mgr ,
Pantoprazol, Plavix , Reminyl , Simvastatina , Sumial Seroquel 100 mgr
19960171;2012-02-03;2/3/2012;2/3/2012;Alprazolam 0.25 mgr, Dormicum 7.5 mgr,
Esertia 10 mgr , Pantoprazol , Plavix, Reminyl 24 mgr , Seroquel 200 1-1-1,
Sinvastatina 20 mgr 1-0-0, Ebixa 20 mgr 1-0-0
19960171;2012-11-08;11/8/2012;11/8/2012;Alprazolam 0.25 mgr, Dormicum 7.5
mgr,Mirtazapina 10 mgr , Pantoprazol , Plavix,Seroquel 100 1-1-1,
Sinvastatina 20 mgr 1-0-0, Ebixa 20 mgr 1-0-0, Keppra 250 1-0-1
19960171;2013-06-27;6/27/2013;6/27/2013;Alprazolam , Dormicum , Duphalac ,
Ebixa 20 mgr , Keppra 250 mgr , Misrtazapina 30 mgr , Pantoprazol , Plavix,
Seroquel 100 mgr 1-1-1, Sinvastatina
19960171;2014-07-01;7/1/2014;7/1/2014;Adiro 100 mgr , Alprazolam 0.25 mgr ,
```

```
Dormicum 7.5 mgr , Duphalac , Keppra 250 mgr y 500 mgr , Pantoprazol 20 mgr
, Seroquel 100 mgr 1-1-1, Sinvastatina 10 mgr
```

lo convertimos con,

```
[osotolongo@brick03 medicacion]$ awk -F";" '{print $1";"$2";"$5}'
seguimientos.csv | sed 's/-//g;s/-//g' > seguimientos.db
```

que es basicamente lo mismo pero seleccionando otras columnas,

```
[osotolongo@brick03 medica]$ head seguimientos.db
Interno;FechaSeguiment;Receta_SIRE_SN
19960014;19981217;
19960044;20050314;igual: tegretol:1/2-0-0
19960068;19990713;
19960171;20100331;Reminyl 24 mgr LR 1-0-0, Axura 20 mgr, Omeprazol 20 mgr
1-0-0, Pazital, Nerdipina 1-0-1, Prevencor 0-1-0,  Sumial 10 mgr, Sedotime
0-0-1 , Plavix 0-1-0.
19960171;20110329;Dormicum , Esertia 10 mgr , Pantoprazol, Plavix , Reminyl
, Simvastatina , Sumial Seroquel 100 mgr
19960171;20120203;Alprazolam 0.25 mgr, Dormicum 7.5 mgr, Esertia 10 mgr ,
Pantoprazol , Plavix, Reminyl 24 mgr , Seroquel 200 1-1-1, Sinvastatina 20
mgr 1-0-0, Ebixa 20 mgr 1-0-0
19960171;20121108;Alprazolam 0.25 mgr, Dormicum 7.5 mgr,Mirtazapina 10 mgr ,
Pantoprazol , Plavix,Seroquel 100 1-1-1, Sinvastatina 20 mgr 1-0-0, Ebixa 20
mgr 1-0-0, Keppra 250 1-0-1
19960171;20130627;Alprazolam , Dormicum , Duphalac , Ebixa 20 mgr , Keppra
250 mgr , Misrtazapina 30 mgr , Pantoprazol , Plavix, Seroquel 100 mgr
1-1-1, Sinvastatina
19960171;20140701;Adiro 100 mgr , Alprazolam 0.25 mgr , Dormicum 7.5 mgr ,
Duphalac , Keppra 250 mgr y 500 mgr , Pantoprazol 20 mgr , Seroquel 100 mgr
1-1-1, Sinvastatina 10 mgr
```

Ahora, creo que lo correcto seria unir estas dos bases de datos y procesarlo todo junto.

```
[osotolongo@brick03 medica]$ tail -n +2 seguimientos.db >
seguimientos_noheader.db
[osotolongo@brick03 medica]$ cat anamnesi.db seguimientos_noheader.db >
todo_medicamentos.db
```

Y ahora tenemos un solo archivo con todos los datos que necesitamos para procesar.

Limpiando y Haciendo las reglas

Limpieza

La manera mas sencilla de hacer el parser es en Perl por varias razones,

1. Las expresiones regulares permiten una limpieza profunda de los datos con casi nada de codigo y a una velocidad razonable
2. Los hashes son muy rapidos para procesar las reglas
3. Existe un modulo (*Text::Levenshtein::XS*) precompilado (rapido) con el calculo de la distancia entre palabras
4. Es muy sencillo insertar un envio de email al final del script

No medicamentos: Como la mayor parte de la limpieza la voy a hacer con expresiones regulares, voy a tomar el archivo *stopwords_orange.txt* que contiene un grupo grande de cadenas de caracteres que **no** son medicamentos y voy a quitar todos lo numeros. Tras esto, quito las lineas que queden en blanco o que esten repetidas.

```
$ sed 's/[0-9]//g;/^[[:space:]]*$/d' ../stopwords_orange.txt | uniq > toremove.list
```

Ahora, este archivo lo necesito de input para el parser. Lo voy a convertir en un *array* y despues solo tengo que quitar estas palabras cada vez que las encuentre. O algo parecido. Asi que dentro del codigo debo incluir,

```
#Leo las palabras a borrar
open ADF, "<toremove.list";
chomp (my @remove = <ADF>);
close ADF;
```

OK, vamos a leer el archivo de datos,

```
open IDF,"<$db" or die "Could not open input file\n";
while(<IDF>){
        if(/^(\d+);(\d+);(.*)$/){
                my ($interno, $fecha, $free) = /^(\d+);(\d+);(.*)$/;
```

y al ultimo campo le cambiamos acentos, dieresis, etc por la letra simple y le quitamos los caracteres no alfanumericos y lo metemos en un *array*, tomando los espacios como separador de valores,

```
                $free = unidecode($free);
                $free =~ s/\W/ /g;
                my @afree = split / /, $free;
```

A cada elemento del array le hacemos una limpieza en este orden,

1. si empieza con numeros nos quedamos con lo que haya despues de los numeros,
2. si tiene numeros dentro, nos quedamos con lo que haya antes de los numeros,
3. por si acaso, borramos todos los numeros que queden,
4. si queda algo, la longitud es mayor que 2 y no esta en el array de palabras a quitar,
    1. cambiamos todo de mayusculas a minusculas
    2. lo añadimos al array de palabras validas
    3. lo contamos

```
                my @nonfree;
                foreach my $token (@afree){
                        if( $token =~ /^\d+.*$/) { $token =~ s/^\d+//g;}
```

```
                        if( $token =~ /^.+\d.*$/){ $token =~ s/\d.*//g; }
                        $token =~ s/\d//g;
                        if($token and length($token)>$wsize and not grep
/($token)/,@remove){
                                $token =~ tr/[A-Z]/[a-z]/;
                                push @nonfree, $token;
                                unless (exists($meds{$token})){
                                        $meds{$token} = 1;
                                }else{
                                        $meds{$token}++;
                                }
                        }
                }
```

cuando terminamos con la linea, guardamos el array de palabras validas correspondiente

```
                $visita{$interno}{$fecha} = [ @nonfree ];
```

lo que me ha quedado es una base de datos supuestamente limpia aunque aun deberia contener palabras que no son medicamentos. Voy a escribirla a disco,

```
my $parsed = 'parsed_meds.csv';
open ODF,">$parsed" or die "Could not open $parsed for writing\n";
foreach my $interno (sort keys %visita){
        foreach my $date_of (sort keys %{$visita{$interno}}){
                print ODF "$interno;$date_of;",join(",", sort
@{$visita{$interno}{$date_of}}),"\n";
        }
}
```

Reglas

Las palabras que se repiten mas de **X** veces van a ser las cabeceras de las reglas. Voy a seleccionarlas,

```
my $repeat = 12;
foreach my $med (sort keys %meds){
        if ($meds{$med} > $repeat){ push @medkw, $med; }
}
```

Y ahora, voy a calcular la distancia de cada palabra a las cabeceras de reglas y la voy a poner bajo la mas cercana. Las palabras que no entren bajo ninguna regla las pondre en un *array* aparte.

```
my $dtresh = 5;
my %medgroups;
my @alonemeds;
foreach my $medword (sort keys %meds){
        my $mindist = 1000;
        my $keyguide = "";
```

```
        foreach my $medkey (@medkw){
                my $dist = distance($medword, $medkey, $dtresh);
                if ((defined $dist) && ($dist < $mindist)) {
                        $mindist = $dist;
                        $keyguide = $medkey;
                }
        }
        if ($keyguide) {
                push @{$medgroups{$keyguide}}, $medword;
        }else{
                push @alonemeds, $medword;
        }
}
```

Y ahora guardo todo esto a disco,

```
my $ofile = 'med_rules.txt';
my $obfile = 'med_no_rules.txt';
open ODF, ">$ofile" or die "Could not create file\n";
foreach my $mgroup (sort keys %medgroups){
        print ODF "$mgroup: ",join("|", sort @{$medgroups{$mgroup}}), "\n";
}
close ODF;
open ODF, ">$obfile" or die "Could not create file\n";
foreach my $med (@alonemeds){
        print ODF "$med\n";
}
close ODF;
```

Ahora, las reglas han de revisarse y editarse a mano. Esto no hay manera de evitarlo pero solo ha de hacerse una vez. La proxima vez que se haga el analisis se usara el mismo archivo de reglas.

Reglas de CIMA

https://cima.aemps.es/cima/publico/nomenclator.html

Me bajo la lista de medicamentos comercializados en España y la convierto a CSV. Luego hago esto:

```
[osotolongo@brick03 medicaion_dev]$ awk -F";" '{print $2}' Medicamentos.csv
| tail -n +2 | sed 's/\// /g' |awk '{print $1}' | tr '[:upper:]' '[:lower:]'
| sort | uniq > medicamentos.list
```

O si quiero tener los principios activos,

```
[osotolongo@brick03 medicaion_dev]$ awk -F";" '{print $2";"$8";"$9}'
Medicamentos.csv | tail -n +2 | sed 's/\// /g' | tr '[:upper:]' '[:lower:]'
| sort | uniq | sed 's/\.//g' > medicamentos_pa.list
[osotolongo@brick03 medicaion_dev]$ sed 's/\([^
```

```
]*\).*;\(.*\);\(.*\)/\1;\2;\3/' medicamentos_pa.list | uniq > mpa.list
```

Aplicando las reglas

Aqui he de leer tres cosas antes que nada. Primero, las reglas que hemos construido. Estas las voy a asignar a un hash de arrays,

```
my %wrules;
open RDF, "<$rules_file" or die "No rules file";
while (<RDF>){
        (my $rkey, my $rvalue) = /^(\w*): (.*)$/;
        chomp $rvalue;
        my @lpat = split /\|/, $rvalue;
        %{$wrules{$rkey}} = map {$_ => 1} @lpat unless !$rkey;
}
close RDF;
```

Luego, la lista de principios activos de CIMA.

```
my %cima;
open ADF, "<$mpa";
while (<ADF>){
        my ($med, $pal, $pan) = /^(.*);(.*);(.*)$/;
        my @pas = split /, /, $pal;
        for (@pas) {s/\s+/_/g;}
        $cima{$med} = [ @pas ];
}
close ADF;
```

Ahora cargo base de datos ya revisada, que si todo va bien sera el output del script anterior.

```
my $dbfile = 'parsed_meds.csv';

my %visits;
my %meds;
open IDF, "<$dbfile" or die "Database file do not exists\n";
while(<IDF>){
        my ($pid, $vid, $ldata) = /^(\d+);(\d+);(.*)$/;
```

Pero voy a ir llenando el hash de output (*%visits*) a medida que leo. Primero meto toda la medicacion de la visita en un array,

```
        if($ldata){
                my @mlist = split /,/, $ldata;
```

y ahora, para cada elemento de este array, recorro las reglas, buscando si este termino existe dentro de alguna regla.

```
                foreach my $med (@mlist){
                        if ($med){
                                foreach my $rkey (sort keys %wrules){
                                        if(exists($wrules{$rkey}{$med})){
                                                foreach my $pa
(@{$cima{$rkey}}){
                                                        $meds{$pa} = 1;
$visits{$pid}{$vid}{$pa} = 1;
                                                }
                                                last;
                                        }
                                }
                        }
                }
```

Cuando encuentro este elemento en las reglas, marco tods los ID de principios activos correspondientes a este medicamento como existentes en esa visita y ademas los sumo a las variables que he de escribir. Si no se encuentra este termino, se concluye que no es ningun medicamento (pues no esta en ninguna regla) y no se hace nada.

Una vez llenados los medicamentos de cada visita hemos de escribirlo en forma matricial. Primero escribimos los headers,

```
 open ODF, ">$ofile" or die "Could not open file !!!!!!\n";
print ODF "Interno,Fecha_visita";
foreach my $med (sort keys %meds){
        print ODF ",$med";
}
print ODF "\n";
```

Y ahora llenamos los valores correspondientes a cada visita con un 1, caso de existir y un 0 en caso contrario.

```
foreach my $pid (sort keys %visits){
        foreach my $vid (sort keys %{$visits{$pid}}){
                print ODF "$pid,$vid";
                foreach my $med (sort keys %meds){
                        if(exists($visits{$pid}{$vid}{$med})){
                                print ODF ",1";
                        }else{
                                print ODF ",0";
                        }
                }
                print ODF "\n";
        }
}
close ODF;
```

TADA!