Serveur Web de Déconversion UNL->FR

Rapport Technique UNL

Introduction

Pour répondre aux nouvelles exigences du projet UNL, il fallait implémenter un serveur de déconversion accessible par Common Gateway Interface (CGI). Le déconvertisseur UNL programmé par l'équipe de l'IAS pouvait alors utiliser directement cette CGI en indiquant son adresse pour la déconversion du français. N'ayant eu aucune indication préalable en ce qui concernait l'architecture de notre serveur, nous avions implanté une interface par courrier électronique. Nous avions donc ajouté un script CGI qui simplement transformait la requête en courrier électronique et l'envoyait à notre serveur.

Cette solution présentait l'avantage de ne pas modifier la partie existante de notre serveur. Par contre, cette architecture présentait quelques défauts. Il était finalement impossible d'utiliser la CGI pour interroger le serveur. Cette CGI devait gérer une boite aux lettres de communication avec notre serveur de déconversion. Elle manquait de robustesse et de stabilité. Elle ne pouvait pas par exemple traiter plus de 99 requêtes simultanément. D'autre part, la communication par courrier électronique ralentissait considérablement le temps global de déconversion.

Lors d'une recherche sur la Toile, j'ai trouvé un protocole d'interrogation de dictionnaires fonctionnant par TELNET. Le Dictionary Server Protocol (DICT) est une transaction TCP basée sur un protocole de requête/réponse qui permet à un client d'accéder à des entrées de dictionnaires. La RFC 2229 le décrivant est disponible à l'adresse suivante : http://www.dict.org/rfc2229.txt.

Nous avions donc pensé modifier l'architecture de notre serveur de façon à ce que la conmmunication entre la CGI et le serveur ne se fasse plus par courrier électronique mais par TELNET/DICT.

Architecture générale

L'utilisateur entre son graphe dans une forme HTML. Le graphe est envoyé à la CGI. qui tourne sur un serveur UNIX.

La CGI tourne sur un serveur UNIX. Elle ouvre une connection TELNET/DICT avec le serveur de déconversion et lui envoie le graphe.

Le serveur de déconversion tourne sur Macintosh. Il transforme le graphe en arbre et envoie l'arbre par courrier électronique au serveur de génération.

Le seeveur de génération ARIANE tourne sur IBM. Lorsqu'il a généré le texte français, il le renvoie au serveur de déconversion par courrier électronique.

Le serveur de déconversion renvoie à son tour le texte français par TELNET/DICT à la CGI qui ferme alors la connexion TELNET/DICT.

La CGI renvoie finalement le texte français à l'utilisateur sous forme HTML.

Interface HTML

Pour entrer son graphe UNL, l'utilisateur utilise soit l'outil de déconversion programmé par l'équipe de l'IAS soit une page HTML avec une forme intégrée. Deux pages HTML sont accessibles sur la Toile à partir d'un navigateur :

- La page http://clips.imag.fr/geta/mathieu.mangeot/unl/unldev.html sert au développement. L'utilisateur peut entrer des paramètres qui sont les mêmes que ceux de l'ancienne version par courier électronique du serveur. L'utilisateur pourra par exemple ne demander que l'arbre (TREE) ou demander un numéro de version spécial du serveur de génération sur IBM.
- La page http://clips.imag.fr/geta/mathieu.mangeot/unl/unl.html sert au public. L'utilisateur entre son graphe et attend le résultat. Il ne peut pas spécifier de paramètres.

Dans l'outil de déconversion de l'équipe de l'IAS et dans les pages HTML, il faut spécifier l'adresse de la CGI. La page HTML unl.html est disponible en annexe.

CGI Perl

La CGI est écrite en script Perl. Elle est exécutée sur un serveur UNIX. Elle est accessible à l'adresse suivante : http://clips.imag.fr/geta/cgi-bin/unl.pl et disponible en annexe.

Cette CGI reçoit les paramètres soit des formes HTML incluses dans les pages HTML indiquées plus haut, soit de l'outil de déconversion de l'équipe de l'IAS. Elle ouvre ensuite une session telnet selon le protocole DICT avec le serveur de déconversion installé sur un macintosh. Par telnet, elle envoie le graphe UNL reçu ainsi que les paramètres éventuels. Elle attend ensuite la réponse du serveur. Cette réponse est ensuite renvoyée à l'utilisateur.

La CGI a besoin de plusieurs paramètres pour être configurée :

•l'adresse du serveur de déconversion : `$host = "dunia.imag.fr";`

•le port utilisé pour la session TELNET/DICT : `$port = 2628;`

Par défaut, j'ai pris le port recommandé pour le protocole DICT par la RFC 2229.

•le délai d'attente de la réponse de la part du serveur de déconversion :

```
$timeout = 150;
```

Ce délai est en secondes. Il faut que le temps de communication entre le serveur de déconversion sur le Macintosh et le serveur de génération sur IBM ainsi que le temps de génération sur IBM soient inclus dans ce délai.

Serveur de déconversion

Le serveur de déconversion est programmé en Macintosh Common LISP. Il est composé de 2 processus principaux : un processus qui écoute sur le port TELNET/DICT et un processus qui scrute une boîte aux lettres.

Lorsque le processus qui écoute sur le port TELNET/DICT reçoit une demande de communication de la part d'un client, il crée un processus fils qui s'occupera de traiter la requête.

Pour l'instant, seules ces commandes sont implémentées :

```
AUTH username password          guess !!

CLUW uw                         b-find-closest-uw

HEAD headword                   b-find-headwords

HELP                            display this text

QUIT                            close connection

TRAN {parameters}               [UNL] unltext [/UNL]

UWTR uw                         look up uw in database
```

Il est possible d'utiliser un identifiant et un mot de passe pour se connecter. Cependant, cette procédure n'est pas mise en place pour simplifier le traitement.

La CGI utilisera la commande TRAN et enverra tous ses paramètres à la suite. La fin de la commande doit impérativement être terminée par la chaîne de caractères "[/unl]" car le serveur attend cette châine pour exécuter la commande.

Le processus fils traite la requête et renvoie la réponse au client. Dans le cas de la commande TRAN, le processus traite la requête comme il le faisait dans la version précédente avec le courrier électronique. Il analyse le graphe UNL, en construit un objet Common Lisp Object System (CLOS). Les noeuds du graphe sont traduits en français à l'aide du dictionnaire UNL-français chargé en mémoire. Le graphe est ensuite transformé en arbre.

Si le client a demandé un arbre, le processus renvoie directement l'arbre au client. Sinon, il envoie l'arbre en courrier électronique au serveur de génération sur IBM. Il boucle en attendant la réponse de celui-ci. Dès qu'elle arrive, elle est aussitôt retransmise au client par TELNET/DICT. Le client ferme alors la connection et le processus meurt. Si une erreur quelconque intervient lors du traitement de la requête, un message d'erreur est envoyé au client en guise de réponse. Celui-ci ferme alors la connection et le processus meurt.

Le processus qui scrute une boîte aux lettres attend les réponses qui proviennent du serveur de génération sur IBM. Lorsqu'une réponse arrive, elle est passée au processus qui l'attendait pour finir le traitement de la requête.

Ce serveur est lui aussi paramétrable. Un panneau de configuration est acessible par le menu principal. On peut indiquer entre autres le numéro de port sur lequel on écoute les connections TELNET/DICT. Par défaut, celui-ci est le même que celui de la RFC 2229 : le port 2628.

Serveur de génération

Le serveur de génération tourne sur IBM. Il est programmé en ARIANE. Il reçoit par courrier électronique des arbres représentant les graphes UNL dont les noeuds ont été traduits en français. Pour chacun de ces arbre, il génère un texte français et le renvoie à l'expéditeur du courrier électronique.

Conclusion

Cette architecture est plus robuste que la précedente car elle crée un processus indépendant pour chaque requête. Elle est aussi plus rapide car elle remplace l'utilisation du courrier électronique par une connection directe TELNET/DICT entre la CGI et le serveur de déconversion.

Pour améliorer encore la rapidité du traitement global d'une requête, il me semble très intéressant de pouvoir remplacer l'utilisation du courrier électronique par une autre connection TELNET entre le serveur de déconversion sur Macintosh et le serveur de génération sur IBM. Des travaux sont en cours sur IBM pour pouvoir implémenter ce genre de service.

Annexe 1 : Page HTML

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2//EN">
<!--This file created 9/10/98 19:27 by Claris Home Page
version 2.0-->
<HTML>
<HEAD>
   <TITLE>UNL-server</TITLE>
   <META NAME=GENERATOR CONTENT="Claris Home Page 2.0">
   <X-SAS-WINDOW TOP=42 BOTTOM=845 LEFT=4 RIGHT=612>
</HEAD>
<BODY BGCOLOR="#AAAAAA" LINK="#0000ED" ALINK="#FE0000"
VLINK="#551A8A" background="marmback.jpg">
<FORM action="http://clips.imag.fr/cgi-
bin/geta/unl/unl.pl" method="POST"
enctype="application/x-www-form-urlencoded">

<P><TABLE  CELLPADDING=3 WIDTH="100%">
   <TR>
      <TD align=LEFT>
         <H1><CENTER>DECO gate - Acc&egrave;s au
         d&eacute;convertisseur
fran&ccedil;ais</CENTER></H1>
      </TD></TR>
</TABLE><INPUT TYPE="hidden" NAME="user" VALUE="gilles">
<INPUT TYPE="hidden" NAME="pass" VALUE="xxx">
<INPUT TYPE="hidden" NAME="lang" VALUE="fr"></P>

<P><CENTER><B>Obtenir la traduction fran&ccedil;aise
de :<BR>

<TEXTAREA NAME="data" ROWS=24
COLS=82></TEXTAREA></B></CENTER></P>

<P><CENTER><INPUT TYPE="submit" NAME="Submit"
VALUE="Deconvertir"><BR>


<HR>

<HR>
<B><I>Des commentaires ? envoyez moi un courriel
&agrave; :
<A HREF="mailto:Mathieu.Mangeot@imag.fr">
</I>Mathieu.Mangeot@imag.fr</A> <I>-</I></B></CENTER></P>
</FORM>
</BODY>
</HTML>
```

Annexe 2 : Script CGI

```
#!/usr/local/bin/perl

$host = "dunia.imag.fr";
$port = 2628;

## il faut compter le temps d'un aller retour par mail
## avec l'IBM pour marseille , c'est suffisant mais pour
## Durian ... je ne crois pas
$timeout = 150;

&html_header("UNL-French Deconverter Server Answer");

############################################################
## decoding the request

&get_request;
$user = $rqpairs{"user"};
$pass = $rqpairs{"pass"};
$lang = $rqpairs{"lang"};
$data = $rqpairs{"data"};

############################################################
## initialisations

if (!$data) {
      print "There is no data posted\n";
      die;
}

$data =~ s/\n//g;

use Net::Telnet ();

$prompt = '/\]>/';

$t = new Net::Telnet (Port => $port,
                      Errmode => "return",
                      Prompt => $prompt,
                      Timeout => 10,);


############################################################
## main

$message = "TRAN {} [UNL][D]$data \[\/D][\/UNL]";


$ok = $t->open($host);

if ($ok) {

# waiting for the confirmation of the connection
    ($lines) = $t->waitfor('/220 .*$/');

    $lines = "";
```

```
# sending the UNL text
    $t->print($message);

# waiting for the answer until $timeout
    ($lines) = $t->waitfor(Match => '/250 .*$/',
                           Timeout => $timeout,);

    if (!$lines) {  ## connection timed out
        print "\n<br>" . $t->errmsg() . "\n<br>";
    }
    else {    ## answer delivered
        $lines =~ s/</&lt;/g;
        $lines =~ s/>/&gt;/g;
        $lines =~ s/\n/<br>/g;
        print $lines;

# ending connection
        $t->print("QUIT");
        $t->waitfor('/221 .*$/');
    }
    $t->close;
}
else {
    print "\n<br>" . $t->errmsg() . "\n<br>";
}

############################################################
## fin

&html_end;

sub get_request {

# Subroutine get_request reads the POST or GET form
# request from STDIN into the variable  $request, and
# then splits it into its name=value pairs in the
# associative array %rqpairs. The number of bytes is
# given in the environment variable CONTENT_LENGTH which
# is automatically set by the request generator.
# Encoded HEX values and spaces are decoded in the values
# at this stage.
# $request will contain the RAW request. N.B. spaces and
# other special characters are not handler in the name
# field.

    if ($ENV{'REQUEST_METHOD'} eq "POST") {
        read(STDIN, $request, $ENV{'CONTENT_LENGTH'});
    } elsif ($ENV{'REQUEST_METHOD'} eq "GET" ) {
        $request = $ENV{'QUERY_STRING'};
    }
    %rqpairs = &url_decode(split(/[&=]/, $request));
}
sub url_decode {
#       Decode a URL encoded string or array of strings
#               + -> space
#               %xx -> character xx

    foreach (@_) {
```

```
        # Un-Webify plus signs and %-encoding
        tr/+/ /;
        s/%(..)/pack("c",hex($1))/ge;

        # Handle ISO Latin 1 character entities
        s/&Aacute;/Á/;
        s/&Acirc;/Â/;
        s/&Agrave;/À/;
        s/&Aring;/A/;
        s/&Atilde;/A/;
        s/&Auml;/Ä/;
        s/&Ccedil;/Ç/;
        s/&aacute;/á/;
        s/&ouml;/ö/;

 # Stop people from using subshells to execute commands
        s/~!/ ~!/g;
   }
    @_;
}

sub html_header {
# Subroutine html_header sends to Standard Output the
# necessary
# material to form an HTML header for the document to be
# returned, the single argument is the TITLE field.

    local($title) = @_;

print 'Content-type: text/html
<HTML>
<HEAD>
<TITLE>', $title, '</TITLE>
</HEAD>
<BODY  BGCOLOR="#ffffff" TEXT="#000000" LINK="#0000dd"
VLINK="#333388" ALINK="#0
0gg00">
      <!-------------------------------------------->
      <!-- Body                                     -->
      <!-------------------------------------------->
';
}

sub html_end {
    print "\n<\/BODY>\n<\/HTML>\n";
}
1;
```