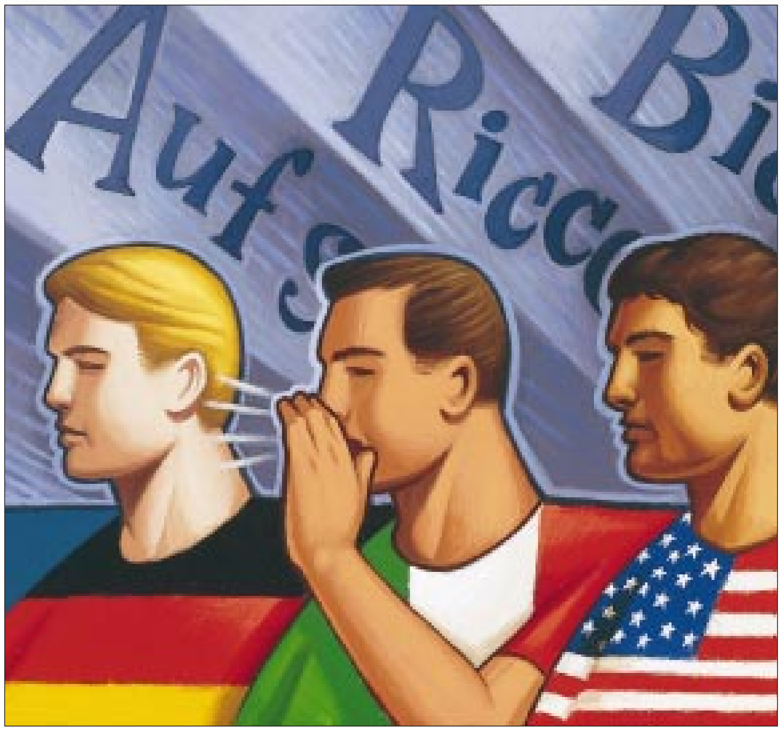
by Jeffreys Copeland and Haemer



ALEX GROSS

"Therefore is the name of it called Babel; because the Lord did there confound the language of all the earth."
– Genesis 11:9

"The Babel fish is small, yellow and leechlike, and probably the oddest thing in the Universe… The practical upshot of all this is that if you stick a Babel fish in your ear you can instantly understand anything said to you in any form of language."
– Douglas Adams, *Hitchhiker's Guide to the Galaxy*

Babelfish

For years, we have been writing, in this magazine and elsewhere, about internationalization: how to write programs that are multilingual.

Unfortunately, we are not yet as multilingual as our software. Haemer recently found the following note in his incoming email:

Sehr geehrter Herr Dr. Haemer!
Herzlichen Dank fuer Ihre Einladung am Donnerstag, die ich dankend annehme Wie Sie vorschlagen, werde ich die Kunstwerke an Ihren Waenden studieren. Dann koennen wir uns auch weiter ueber solche Themen wie Religion, Literatur und Philosophie unterhalten. Ich bringe Salat. Bitte schicken Sie mir directions to your Haus.
Herzlich, Ana

Haemer speaks no German. What to do, what to do?

If you've never clicked on the little [Translate] link in AltaVista, this is the time to try it out. Doing so takes you to `http://babelfish.altavista.com`, home of a server that will mechanically translate some natural languages into others. Babelfish offers machine translation between English and a handful of common, European languages: French, German, Italian, Portuguese and Spanish.

Machine translation has a long, mixed history. The bottom line is the translations aren't perfect, and some of them are downright silly. What impresses us after a little use is how often Babelfish's translations are good enough.

Here's what Babelfish did with Ana's message:

Dear Dr. Haemer!
Cordial thanks for your invitation on Thursday, which I assume thanking. As you suggest, I will study the works of art at your walls. Then we can converse also further about such topics such as religion, literature and philosophy. I bring salad. Please you send directions to to me your house.
Cordially, Ana

Imperfect, but completely intelligible. It would be a pity to miss such wonderful email because of our linguistic limitations or because we demand perfect translations.

A joint venture of Systran and AltaVista, Babelfish is named after Douglas Adams' Babel fish from *Hitchhiker's Guide to the Galaxy*. It's mostly used to make sense of Web pages that would otherwise be gibberish, but it's a general-purpose tool.

Babelfish: Not Just for Browsers Anymore

Unfortunately, but not surprisingly, the AltaVista interface is tied to a browser, which limits you to typing at it. If you're interested in both programming and human languages, as we are, there are a lot of fun things you can imagine trying with a programmatic interface to a tool like this. However, writing Web clients from scratch can require a lot of ad hoc, trial-and-error work (see "A Simple Web Script," April 1999, Page 44, `http://sw.expert.com/C9/SE.C9.APR.99.pdf`).

Luckily, there is now a Perl module on the Comprehensive Perl Archive Network, or CPAN (`http://www.perl.com/CPAN/`), to help. `WWW::Babelfish` is a module specifically designed to let you write Web clients for the Babelfish server. It took us only a few minutes to produce a working, general-purpose translation script. A few more iterations brought us to the script shown in Listing 1.

We'll mostly let the code speak for itself. The documentation that comes with Babelfish is very clear, and this column has so much code that we're cramped for space. The third line points the script at our own, local version of `WWW::Babelfish`. We had to add the line

```
$ua->proxy(['http', 'ftp'] => $ENV{http_proxy});
```

to let us get to the Web through proxy servers.

We Play Telephone

So, let's try it out.

In the game of "telephone," a sentence is whispered from person to person around a circle. When it completes a circuit, the original phrase is contrasted with what it has turned into.

Imagine a game of telephone at the European Commission headquarters in Brussels, where each player has a different native language and most of the transmission noise is translation error. We can simulate this by setting up a pipeline of translation programs. Because Babelfish only provides translations to and from English, every other speaker will have to be an Anglophone. (Alternatively, imagine that all the whispering is done in English, but that each speaker must translate the phrase that comes in his right ear, first from English into his native language and then back to English, before passing it on to the person on his left.)

Here's an example:

```
#!/bin/sh
# Continental telephone
# $Id: teleEurope,v 1.1 1999/08/09 16:26:36 jsh Exp $
whisper()   {
   ana -o $1  | ana -i $1 | tee /dev/stderr
}

echo my hovercraft is full of eels |
   whisper English |
   whisper French |
   whisper Portuguese |
   whisper Italian |
   whisper German |
   whisper Spanish
```

And here's its output:

```
my hovercraft is full of eels

my hovercraft is full with eels

mine hovercraft is full with conger-eels

Hovercraft of the mine is full with the gronghi

Air cushion vehicle of the pit is full with gronghi
```

Listing 1. A Simple Babelfish Client

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

use strict;
use lib ".";        # hack, cough
use WWW::Babelfish;
use Getopt::Std;
my $options = "[-i input_language | -o output_language] [filename ...]";
my $usage = "usage: $0 $options";

sub get_langs   {
   use vars qw($opt_o $opt_i);
   getopts "i:o:" or die $usage;
   die $usage if ($opt_o && $opt_i);
   my ($in, $out) = ($opt_i || "English", $opt_o || "English");
}

my ($in, $out) = get_langs;
my $obj = new WWW::Babelfish( 'agent' => 'Mozilla/8.0' );
die "Babelfish server unavailable\n" unless defined $obj;

my @languages = $obj->languages;
die "source language $in must be in @languages\n"
   unless grep /$in/, @languages;
die "destination language $out must be in @languages\n"
   unless grep /$out/, @languages;

$/ = undef;
my $translation = $obj->translate(source=>$in, destination=>$out, text=><>);

die "Could not translate: " . $obj->error unless defined $translation;
print "$translation\n";

=head1 NAME

ana - Simple Babelfish client, for notes from Ana

=head1 SYNOPSIS

ana [-i input_language | -o output_language] [files]
```

Continued on Page 40

```
=head1 DESCRIPTION
=over 2

B<ana> uses babelfish to translate from one language to another.
Default language for each is English.

=back

=head1 OPTIONS AND ARGUMENTS

=over 8

=item I<-i>

input language

=item I<-o>

output language

=item I<filename ...>

files to translate (default: stdin)

=back

=head1 AUTHOR

Jeffrey S. Haemer <jsh@usenix.org>

=head1 SEE ALSO

   perl(1) WWW::Babelfish(3)

=cut
```

```
The vehicle of pneumatic shock absorber of the
hollow is full with gronghi
```

("My hovercraft is full of eels" is used in the `WWW::Babelfish` documentation, and is taken from Monty Python's "Hungarian Phrasebook" sketch.)

But why stop there? Looking for Babelfish-related news articles on DejaNews (`http://www.dejanews.com`), we found the following wonderful post from David Chess at IBM Research:

```
> Subject: Babelfish invariance
> From: chess@us.ibm.com (David M. Chess)
> Date: 1999/05/27
> Newsgroups: alt.hackers
> The first thing everyone does with a translator
> like Babelfish (http://babelfish.altavista.digital.
> com/) is to translate something from one's native
> tongue into some other language, and then back
> again, to see what happens. It's only a slight
> stretch to *continue* this process until you get
> to a fixed point of the transform (the resulting
> string is the same as the last one you put in),
> or a cycle (the resulting string is the same as
> the one you put in N steps back). A string in
> language A which, when translated into language
> B by Babelfish and the result translated back
> into A, yields A again, is said to be "Babelfish
> invariant".
> [...]
```

Further on in the posting, David says he handcrafted a client to play with this idea. Lazily, we tried our hands at the same thing with `WWW::Babelfish`:

```
#!/usr/local/bin/perl -w
#$ID: telephone,v 1.3 1999/09/01 20:13:52 jeff Exp jeff $

use strict;
use lib ".";        # hack, cough
use WWW::Babelfish;
use Getopt::Std;
my ($obj, $in, $phrase);
my $optionsa  = "[-c] [-v]";
my $optionsb1 = "[-s language_spoken ";
my $optionsb2 = "| -t language_of_thought]";
my $optionsb  = $optionsb1 . $optionsb2;
my $optionsc1 = "[-n cycles]";
my $optionsc2 = "[filename | -e expression]";
my $optionsc  = $optionsc1 . $optionsc2;
my $usage = "usage: $0 $optionsa $optionsb $optionsc";
use vars qw($opt_s $opt_t $opt_n $opt_v $opt_e $opt_c);

sub parse_args   {
   getopts "s:t:n:e:vc" or die $usage;
   die $usage if ($opt_s && $opt_t);
   my ($speak, $think) = ($opt_s ||
       "English", $opt_t || "English");
   $speak = ucfirst lc $speak;
   $think = ucfirst lc $think;
   my $n = $opt_n || 10;
   die unless $n =~ /^\d+$/;

   ($speak, $think, $n);
}

sub xform   {
   my ($s, $d, $in) = @_;
   warn "in = $in\n" if ($opt_v);
   my $out = $obj->translate(source=>$s,
      destination=>$d, text=>$in);
   warn "out = $out\n" if ($opt_v);
   die "Could not translate: $s" .
      $obj->error unless defined $out;
   chomp $out;
   $out;
}

my ($speak, $think, $n) = parse_args;
$obj = new WWW::Babelfish( 'agent' => 'Mozilla/8.0' );
die "Babelfish server unavailable\n"
   unless defined $obj;

my @languages = $obj->languages;
die "Spoken language ($speak) must be in:
   @languages.\n"
   unless grep /$speak/, @languages;
die "Language of thought ($think) must be in:
   @languages.\n"
   unless grep /$think/, @languages;

if ($opt_e)   {
   $phrase = $opt_e;
   die $usage if @ARGV;
}  else   {
   local $/ = undef;
   $phrase = <>;
}
$in = $phrase;

foreach my $t (1..$n)   {
   my $out = xform $speak, $think, $in;
   $out = xform $think, $speak, $out;
   if (lc $in  eq lc $out)   {
        chomp $in;
        print "$t\t" if $opt_c;
        print "$in\n";
        exit;
   }
   $in = $out;
}

die qq("$phrase"\n\thas become\n"$in"\n);

=head1 NAME

telephone - simulates the game of "telephone"

=head1 SYNOPSIS

telephone [-c] [-nI<n>] [-v] [files | -e expression]
[-t thought language | -s spoken langauge]
```

```
=head1 DESCRIPTION

=over 2

B<telephone> simulates the game of telephone. (In the
game of telephone, participants sit in a big circle.
One person whispers a phrase to the person next to
him. That person then whispers what he thought he
heard to the person on the other side, and this
continues around the circle until it gets back to
the originator. The point of the game is to see how
much the phrase changes in transit.

In this program, each simulated participant "thinks"
in one language (say, German) and "whispers" in a
second (say, English). The changes are generated by
one cycle of translating from English to German and
back again. Translation is performed by babelfish.

This process continues through a series of
English->German->English cycles until the phrase
has either become "babelfish invariant" (stable)
or gone around the circle.

=back

=head1 OPTIONS AND ARGUMENTS

=over 8

=item I<-v>

verbose

=item I<-t>

language of thought

=item I<-s>

language of speech

=item I<-n>n

number of participants (default: 10)

=item I<-c>

count iterations until stability

=item I<-e>

word or expression to translate

=item I<filename ...>

files to translate (default: stdin)

=back

=head1 AUTHOR

Jeffrey S. Haemer <jsh@usenix.org> and
Jeffrey Copeland <copeland@alumni.caltech.edu>
from a suggestion in alt.hackers
by David M. Chess <chess@us.ibm.com>

=head1 SEE ALSO

   perl(1) WWW::Babelfish(3)

=cut
```

David also says that French is rumored to be the best-supported language. To test this, we wrote a little shell script that plays telephone in several languages:

```
#!/bin/sh
# $Id: multi-tel,v 1.1 1999/08/09 16:26:36 jsh Exp $
# comparison of languages for "telephone"

for i in English French German Italian Portuguese Spanish
do
   echo == $i
   telephone -c -t $i -e "$*"
done
```

and another to exercise it:

```
#!/bin/sh
#! $Id: mtest,v 1.1 1999/08/09 16:26:36 jsh Exp $
# demo of multi-tel

multi-tel My hovercraft is full of eels.
echo
multi-tel Out of sight, out of mind.
echo
multi-tel CITRAN blows dead aardvarks.
```

Here's what we found when we ran it:

```
== English
1       My hovercraft is full of eels.
== French
2       My hovercraft is full with eels.
== German
5       My air cushion, machine pulls up,
        is full from the Aalen.
== Italian
2       My Hovercraft is full of the eels.
== Portuguese
3       Hovercraft of the mine is full of
        conger-eels.
== Spanish
1       My hovercraft is full of eels.
== English
1       Out of sight, out of mind.
== French
3       Out of the sight, spirit.
== German
4       Understand over the sight from out.
== Italian
2       From sight, the mind.
```

```
== Portuguese
3       Except of the sight, it is of the mind.
== Spanish
2       Outside Vista, the mind.

== English
1       CITRAN blows dead aardvarks.
== French
2       CITRAN blows the dead aardvarks.
== German
"CITRAN blows dead aardvarks."
has become
"CITRAN burns continuous aardvarks one of the dead
ones of one."
== Italian
"CITRAN blows dead aardvarks."
has become
"CITRAN jumps the aardvarks has put put out of order
put put put put put put put."
== Portuguese
2       Inoperative CITRAN establish aardvarks.
== Spanish
"CITRAN blows dead aardvarks."
has become
"Aardvarks died to the blowing of CITRAN."
```

We like reading these out loud in thick, stage accents.

The numbers at the beginning of each line are the number of steps to Babelfish invariance for that language. If no stable phrase has been found after 10 steps, the beginning and ending phrases are printed, as with the German, Italian and Spanish translations of the third phrase.

It looks to us like the Spanish translations may be better than the French; however, the German translations are certainly the worst. Because we began this column by showing how useful the German translations are, the Spanish translations must be very good indeed.

We'll leave you with a few questions.

Reader Quiz 1: *Obviously, different input words can produce different translations, but does Babelfish take punctuation into account, or just pass it through, unchanged?*

Also noteworthy is the final phrase's failure to stabilize in several languages. In his posting, David notices that some strings fail to stabilize because the translation goes into an infinite regression. Try this:

```
telephone -v -t french -e pizza
```

Reader Quiz 2: *Can anyone offer a word or phrase that puts telephone into an infinite loop by alternating between two (or more) translations?*

Oh, and the two new other phrases? "Out of sight, out of mind" has a venerable history in machine translation. It is said that an early attempt to translate this phrase to Russian and back returned "Invisible idiot."

Reader Quiz 3: *Can anyone out there provide a real citation for this oft-recited, possibly apocryphal story?*

Reader Quiz 4: *Can anyone tell us what* CITRAN *was, why it blew dead aardvarks, and who originally pointed this out?*

Soon after we finished writing this article, National Public Radio (NPR) ran an item about the Consortium for Speech Translation Advanced Research, or C-STAR, based at Carnegie-Mellon University in Pittsburgh, PA. C-STAR has developed a prototype machine translation system that, by operating in restricted domains–C-STAR's example is a travel agency–can do nearly simultaneous translation from *voice input*. If you have the RealNetworks Inc. RealAudio plug-in for your browser, you can listen to it from the Web page for the July 22nd edition of "All Things Considered," at `http://www.npr.org`.

Summary

In this column, we tried to tie together three topics that we're interested in: programming for the fun of it, the Web and internationalization.

But what about the art on Haemer's walls? If you want to come see it, send him email in English, French, German, Italian, Portuguese or Spanish. Plan to bring salad.

Until next time, happy trails.

Jeffrey Copeland *(*`copeland@alumni.caltech.edu`*) lives in Boulder, CO, and works at Softway Systems Inc. on UNIX internationalization. He spends his spare time rearing children, raising cats and being a thorn in the side of his local school board.*

Jeffrey S. Haemer *(*`jsh@usenix.org`*) works at QMS Inc. in Boulder, CO, building laser printer firmware. Before he worked for QMS, he operated his own consulting firm and did a lot of other things, like everyone else in the software industry.*

Note: The software from this and past Work columns is available at `http://alumni.caltech.edu/~copeland/work` *or alternately at* `ftp://ftp.expert.com/pub/Work`*.*