Preparing to use apertium-transfer-tools

En français

apertium-transfer-tools has an example (http://apertium.svn.sourceforge.net/viewvc/apertium/trunk/apertium-transfer-tools/example/) directory in the source package, which should be the first place to look for information.

The alignment templates we use are similar to Moses' 'Factored Models', if you have experience with those (http://www.statmt.org/moses/?n=Moses.FactoredModels)

I should come clean about something: there's a lot of work involved before you can use a-t-t.

First, you need a bilingual corpus: sentence aligned, one language per file, one sentence per line. I assume that you have those.

It's good, at this point, to make sure you have a clear understanding of Apertium's whole architecture.

Taking my example sentence, and running it through the Alpha testing (http://www.apertium.org/testing/) section with 'Print intermediate representation' checked.

```
Esta es Gloria, mi amiga argentina

lt-proc (morphological analysis mode):
^Esta/Este<prn><tn><f><sg>/Este<det><dem><f><sg>$
^es/ser<vbser><pri><p3><sg>$
^Gloria/Gloria<n><f><sg>/Gloria<np><ant><f><sg>$^,/,<cm>$
^mi/mío<det><pos><mf><sg>$ ^amiga/amigo<adj><f><sg>/amigo<n><f><sg>$
^argentina/argentino<adj><f><sg>/argentino<n><f><sg>$

apertium-tagger:
^Este<prn><tn><f><sg>$ ^ser<vbser><pri><p3><sg>$
^Gloria<np><ant><f><sg>$^,<cm>$ ^mío<det><pos><mf><sg>$
^amigo<n><f><sg>$ ^argentino<adj><f><sg>$

apertium-pretransfer:
^Este<prn><tn><f><sg>$ ^ser<vbser><pri><p3><sg>$
^Gloria<np><ant><f><sg>$^,<cm>$ ^mío<det><pos><mf><sg>$
^amigo<n><f><sg>$ ^argentino<adj><f><sg>$

apertium-transfer:
^Prn<SN><tn><mf><sg>{^this<prn><tn><3><4>$}$
^verbcj<SV><vbser><pri><p3><sg>{^be<vbser><pri><p3><sg>$}$
^ant<SN><f><sg>{^Gloria<np><ant><f><sg>$}$^coma<cm>{^,<cm>$}$
^det_nom_adj<SN><f><sg>{^my<det><pos><sg>$ ^Argentinian<adj>$
^friend<n><3>$}$
```

```
apertium-interchunk:
^Prn<SN><tn><mf><sg>{^this<prn><tn><3><4>$}$
^verbcj<SV><vbser><pri><p3><sg>{^be<vbser><pri><p3><sg>$}$
^ant<SN><f><sg>{^Gloria<np><ant><f><sg>$}$^coma<cm>{^,<cm>$}$
^det_nom_adj<SN><f><sg>{^my<det><pos><sg>$ ^Argentinian<adj>$
^friend<n><3>$}$

apertium-postchunk:
^This<prn><tn><mf><sg>$ ^be<vbser><pri><p3><sg>$
^Gloria<np><ant><f><sg>$^,<cm>$ ^my<det><pos><sg>$ ^Argentinian<adj>$
^friend<n><sg>$

lt-proc (generation mode):
This is Gloria, my Argentinian friend

lt-proc (orthographic correction mode - unused in this example):
This is Gloria, my Argentinian friend
```

a-t-t only generates input to 'apertium-transfer' - everything before that point (and after) needs to be provided first: you need morphological analysers for each language involved - I assume that you're going to use a pair of analysers that we already have.

Also; the rules that a-t-t generates are for the 'transfer only' mode of apertium-transfer: this example uses the chunk mode - most language pairs, unless the languages are *very* closely related, would really be best served with chunk mode. Converting a-t-t to support this is on my todo list, and though doing it properly may take a while, I can probably get a crufty, hacked version together fairly quickly. With a couple of sed scripts and an extra run of GIZA++ etc., we can also generate rules for the interchunk module.

Next, you need probability files for the part-of-speech taggers: Tagger_training, TSX_format

Newer releases of CG (http://beta.visl.sdu.dk/cg3.html) have (partial) support for Apertium's stream format. CG is a much better general purpose tagger than Apertium's, but Apertium's is much faster. Again, the Apertium wiki has some information: Constraint_Grammar, Apertium_and_Constraint_Grammar

We also have some instructions for converting CG to TSX, for tagger training. With a good enough CG grammar, it should be possible to use the 'supervised training' mode of the tagger.

We also need a bilingual dictionary. If they aren't available, we have tools available to help construct them automatically: Crossdics as I mentioned in my article, and ReTraTos which can build Apertium-format dictionaries from the same alignments generated by GIZA++ - the output of this should be manually checked, however, as it can output many questionable entries, particularly with multiword expressions.

The need for the bilingual dictionary seemed a little strange to me at first, but Mikel, Apertium's BDFL, explained that it really helps to reduce bad alignments. This probably means that a-t-t can't generate rules for things like the Polish to English 'coraz piękniejsza' -> 'prettier and prettier', but I haven't checked that yet.

So far, these are all things that are necessary for the translator anyway. Next, there are two specific types of files that are required by a-t-t: an 'atx' file, which specifies lexicalised words, and two 'ptx' files. It should be possible to use the example .atx file that comes with a-t-t after just changing the language identifiers. The .ptx files are used to specify 'mlu's - multiple lexical units. For Spanish, these are verbs with enclitic pronouns ('Dímelo' - 'Say it to me' is analysed as: '^Dímelo/Decir<vblex><imp><p2><sg>+prpers<prn><enc><p1><mf><sg>+lo<prn><enc><p3><nt>/Decir<vblex><imp><p2><sg>+prpers<prn><enc><p1><mf><sg>+prpers<prn><enc><p3><m><sg>$'); in the other direction, "John's dog"[1] becomes "el perro de John" - a simple ptx for Spanish would look like this:

```
<?xml version="1.0" encoding="UTF-8"?>
<posttransfer>
<mlu>
  <lu tags="vblex.*"/>
  <lu tags="prn.enc.*"/>
  <lu tags="prn.enc.*"/>
</mlu>
</posttransfer>
```

and for English, like this:

```
<?xml version="1.0" encoding="UTF-8"?>
<posttransfer>
<mlu>
  <lu tags="n.*"/>
  <lu tags="gen.*"/>
</mlu>
</posttransfer>
```

Generally speaking[1] you can find the relevant tags for mlus by grepping for '<j/>' in the morphological analysers.

Finally(!), you need a modes file; the sample modes file can be used, substituting language abbreviations.

[1] The analysis of this is "^John/John<np><ant><m><sg>$^'s/'s<gen>$ ^dog/dog<n><sg>$" - the '+' is missing here because the analysis broke off at the non-alphabet character ("'").
