Contributing to an existing pair

En français

This is a guide on how to add linguistic data directly to an existing language pair in Apertium. It gets a bit technical – if you just want to notify us of some errors or pass along a word list, please see Contributing. If you get stuck while going through this guide, please get in touch on IRC or the mailing list (see Contact for how to do that).

Apertium has data for many languages pairs. These linguistic data include mainly dictionaries (monolingual and bilingual), structural transfer rules that perform grammatical and other transformations between the two languages involved, and lexical data for the part-of-speech tagger, which is in charge of the disambiguation of the source language text.

For most older released language pairs (the ones in 'trunk'), all the linguistic data are contained in a single directory. For example, `apertium-es-ca` for the Spanish-Catalan pair. For newer language pairs, we separate out the monolingual data (analysers/generators/disambiguators) into individual modules, and keep only the bilingual data (lexical selection, bilingual transfer and transfer rules) in the "pair" directory.

It's also possible that only one side of the pair has been split out into a monolingual directory. When you type "./autogen.sh" in the pair directory, it should tell you if you need to use a monolingual package.

Consider contributing your improved lexical data

If you have successfully added general-purpose lexical data to any of the Apertium language pairs, please consider contributing it to the project so that we can offer a better toolbox to the community. The current preferred way of doing this is by pull request on GitHub.

Alternatively, you can e-mail your data (typically three XML files, one for each monolingual dictionary and another one for the bilingual dictionary) to the maintainer of the language data, or to the public apertium-stuff mailing list.

Contents

The maintainer of a language pair is typically listed in one of the files in that pair, do

```
grep @ configure.ac AUTHORS
```

to look for e-mail addresses. Alternatively, just send it to apertium-stuff.

If you believe you are going to contribute more heavily to the project:

1. Join IRC and introduce yourself on the mailing list! Existing Apertium developers can guide you.
2. If you're new to git/GitHub, take a look at the resources on Using Git.
3. Otherwise, create forks of the GitHub repositories you are developing, and submit pull requests.
4. Eventually, you can request write access for those repositories, or become a member of https://github.com/apertium.

(For more details, see our page on repository permissions).

You should be aware that the data you contribute to the project, once added, will be freely distributed under the current license (GNU General Public License 2+). Make sure the data you contribute is not affected by any kind of license which may be incompatible with the licenses used in this project. No kind of agreement or contract is created between you and the developers. If you have any doubt, or you plan to make a massive contribution, contact Mikel L. Forcada.

Example file layout

Pair separate from monolingual data

For the Urdu-Hindi pair, the apertium-urd-hin pair directory contains:

- apertium-urd-hin.hin-urd.lrx : Lexical selection rules for Hindi to Urdu
- apertium-urd-hin.hin-urd.t1x : Transfer rules for Hindi to Urdu
- apertium-urd-hin.urd-hin.dix : Bilingual dictionary between Hindi and Urdu (goes both ways)
- apertium-urd-hin.urd-hin.lrx : Lexical selection rules for Urdu to Hindi
- apertium-urd-hin.urd-hin.t1x : Transfer rules for Urdu to Hindi
- modes.xml : Pipeline definitions for using and debugging the translator

The monolingual directory apertium-urd contains:

- apertium-urd.urd.dix : Urdu monolingual dictionary (for analysis and generation)
- apertium-urd.urd.rlx : Constraint Grammar rules (for disambiguation)
- modes.xml : Pipeline definitions for using and debugging the analyser and disambiguator

(Many monolingual data directories also contain a compiled tagger file for disambiguation, in the above case it would be named `urd.prob`.)

The apertium-hin directory contains similar files to apertium-urd.

All in one directory

For the Spanish–Catalan pair (apertium-es-ca):

- apertium-es-ca.es.dix : Spanish monolingual dictionary, containing 11,800 entries (as of 17 november 2005)
- apertium-es-ca.ca.dix : Catalan monolingual dictionary, containing 11,800 entries.
- apertium-es-ca.es-ca.dix : Spanish-Catalan bilingual dictionary, containing 12,800 entries (correspondences Spanish-Catalan).
- apertium-es-ca.es-ca.t1x : Structural transfer rules for the translation from Spanish to Catalan.
- apertium-es-ca.ca-es.t1x : Structural transfer rules for the translation from Catalan to Spanish.
- apertium-es-ca.es.tsx : Tagger definition file for Spanish
- apertium-es-ca.ca.tsx : Tagger definition file for Catalan
- apertium-es-ca.post-es.dix : Post-generation dictionary for Spanish, with 25 entries and 5 paradigms (applies when translating from Catalan to Spanish)
- apertium-es-ca.post-ca.dix : Post-generation dictionary for Catalan, with 16 entries and 57 paradigms (applies when translating from Spanish to Catalan)
- directory es-tagger-data : Contains data needed for the Spanish tagger (corpora, etc.)
- directory ca-tagger-data : Contains data needed for the Catalan tagger (corpora, etc.)

(some old pairs from the Apertium 1 days name their transfer rules like apertium-oc-ca.trules-oc-ca.xml)

Adding words to the dictionaries

When extending or adapting Apertium, the most likely operation that will be performed will be to extend its dictionaries. In fact, it will be far more common than adding transfer or post-generation rules.

IMPORTANT: Every time a set of modifications is made to any of the dictionaries, the modules have to be recompiled. Type *make* in the directory where the linguistic data are saved (apertium-es-ca, apertium-es-gl or what may be applicable) so that the system generates the new binary files. If you changed monolingual data, you have to type *make* in directory with the monolingual data. If you want to test the translator, you have to type *make* there as well.

If you want to add a new word to Apertium, you need to add three entries in the dictionaries. Suppose you are working with the Spanish-Catalan pair. In this case, you have to add:

1. an entry in the Spanish monolingual dictionary: so that the translator can analyze ("understand") the word when it finds it in a text, and generate it when translating this word into Spanish.

2. an entry in the bilingual dictionary: so that you can tell Apertium how to translate this word from one language to the other.
3. an entry in the Catalan monolingual dictionary: so that the translator can analyze ("understand") the word when it finds it in a text, and generate it when translating this word into Catalan.

You will need to go to the directory containing the XML dictionaries (for the Spanish-Catalan pair, this is apertium-es-ca) and open with a text editor or a specialized XML editor the three dictionary files mentioned: apertium-es-ca.es.dix, apertium-es-ca.es-ca.dix and apertium-es-ca.ca.dix. The entries you need to create in these three dictionaries share a common structure.

Monolingual dictionary (Spanish)

You may want, for example, to add the Spanish adjective "cósmico", whose equivalent in Catalan is "còsmic". The first step is to add this word to the Spanish monolingual dictionary. You will see that a monolingual dictionary has basically two types of data: paradigms (in the "<pardefs>" section of the dictionary, each paradigm inside a <pardef> element) and word entries (in the main <section> of the dictionary, each one inside an <e> element). Word entries consist of a lemma (that is, the word as you would find it in a typical paper dictionary) plus grammatical information; paradigms contain the inflection data of all lemmas in the dictionary. You can search a particular word by searching the string lm="word" (lm meaning lemma). (The element lm is optional and some other dictionaries may not contain it.) Look at the word entries in the Spanish monolingual dictionary, for example at the entry for the adjective "bonito". You can find it by searching lm="bonito":

```
<e lm="bonito">
  <i>bonit</i>
  <par n="absolut/o__adj"/>
</e>
```

To add a word, you will have to create an entry with the same structure. The part between and contains the prefix of the word that is common to all inflected forms, and the element <par> refers to the inflection paradigm of this word. Therefore, this entry means that the adjective "bonito" inflects like the adjective "absoluto" and has the same morphological analysis: the forms *bonito, bonita, bonitos, bonitas* are equivalent to the forms *absoluto, absoluta, absolutos, absolutas* and have the morphological analysis: adj m sg, adj f sg, adj m pl and adj f pl respectively.

Now, you have to decide which is the lexical category of the word you want to add. The word "cósmico" is an adjective, like "bonito". Next, you have to find the appropriate paradigm for this adjective. Is it the same as the one for "bonito" and "absoluto"? Can you say *cósmico, cósmica, cósmicos, cósmicas*? The answer is yes, and, with all this information, you can now create the correct entry:

```
<e lm="cósmico">
  <i>cósmic</i>
  <par n="absolut/o__adj"/>
</e>
```

If the word you want to add has a different paradigm, you have to find it in the dictionary and assign it to the entry. You have two ways to find the appropriate paradigm: looking in the <pardefs> section of the dictionary, where all the paradigms are defined inside a <pardef> element, or finding another word that you think may already exist in the dictionary and that has the same inflection paradigm as the one to be added. For example, if you want to add the word "genoma", you need to find an appropriate paradigm for a noun whose gender is masculine and forms the plural with the addition of an -s. This will be the paradigm "abismo__n" in our present dictionaries. Therefore, the entry for this new word would be:

```
 <e lm="genoma">
    <i>genoma</i>
    <par n="abismo__n"/>
  </e>
```

In exceptional cases you will need to create a new paradigm for a certain word. You can look at the structure of other paradigms and create one accordingly. For a more detailed description of paradigms and word entries in the dictionaries, refer to section 3.1.2 of the system documentation.

Monolingual dictionary (Catalan)

Once you have added the word to one monolingual dictionary, you have to do the same to the other monolingual dictionary of the translation pair (in our example, the Catalan monolingual dictionary) using the same structure. The result would be:

```
<e lm="còsmic">
   <i>còsmi</i>
   <par n="acadèmi/c__adj"/>
</e>
```

Monolingual dictionary (Galician)

In the case you are trying to improve the XML dictionaries for the Spanish-Galician pair, you will need to go to the directory apertium-es-gl and open with a text editor or a specialized XML editor the three dictionary files apertium-es-gl.es.dix, apertium-es-gl.es-gl.dix and apertium-es-gl.gl.dix. In that case, once you have added the new Spanish word "genoma" to the Spanish monolingual dictionary (apertium-es-gl.es.dix), you have to add the equivalent Galician word "xenoma" to the Galician monolingual dictionary (apertium-es-gl.gl.dix), that is:

```
<e lm="xenoma">
   <i>xenoma</i>
   <par n="Xulio__n"/>
</e>
```

Bilingual dictionary

The last step is to add the translation to the bilingual dictionary. A bilingual dictionary does not usually have paradigms, only lemmas. An entry contains only the lemma in both languages and the first grammatical symbol (the lexical category) of each one. Entries have a left side (<l>) and a right side (<r>), and each language has always to be in the same position: in our system, it has been agreed that Spanish occupies the left side, and Catalan, Galician and Portuguese the right side. Once the "side" of each language has been agreed in a system, it has to be observed all through the dictionaries so that the translation works. With the addition of the lemma of both words, the system will translate all their inflected forms (the grammatical symbols are copied from the source language word to the target language word). This will only work if the source language word and the target language word are grammatically equivalent, that is, if they share exactly the same morphological symbols for all of their inflected forms. This is the case with our example; therefore, the entry you have to add to the bilingual dictionary is:

```
<e>
   <p>
     <l>cósmico<s n="adj"/></l>
     <r>còsmic<s n="adj"/></r>
   </p>
 </e>
```

This entry will translate all the inflected forms, that is, *adj m sg*, *adj f sg*, *adj m pl* and *adj f pl*. It works for the translation in both directions: from Spanish to Catalan and from Catalan to Spanish.

In the case of the Spanish-Galician pair, the following bilingual entry in the Spanish-Galician bilingual dictionary (apertium-es-gl.es-gl) will translate all the inflected forms for the equivalent words genoma/xenoma in both directions, that is, from Spanish to Galician and from Galician to Spanish:

```
<e>
   <p>
     <l>genoma<s n="n"/></l>
     <r>xenoma<s n="n"/></r>
   </p>
 </e>
```

What to do if the word pair is not equivalent grammatically (their grammatical symbols are not exactly the same)? In that case, you need to specify all the grammatical symbols (in the same order as they are specified in the monolingual dictionaries) until you reach the symbol that differs between the source language word and the target language word. For example, the Spanish noun "limón" has masculine gender and its equivalent in Catalan, "llimona", has feminine gender. The entry in the bilingual dictionary must be as follows:

```
 <e>
   <p>
     <l>limón<s n="n"/><s n="m"/></l>
     <r>llimona<s n="n"/><s n="f"/></r>
```

```
    </p>
  </e>
```

A more difficult problem arises when two words have different grammatical symbols and the grammatical information of the source language word is not enough to determine the gender (masculine or feminine) or the number (singular or plural) of the target language word. Take for example the Spanish adjective "canadiense". Its gender is (masculine-feminine) since it is invariable in gender, that is, it can go both with masculine and feminine nouns (hombre canadiense, mujer canadiense). In Catalan, on the other hand, the adjective has a different inflection for the masculine and the feminine (home canadenc, dona canadenca). Therefore, when translating from Spanish to Catalan it is not possible to know, without looking at the accompanying noun, whether the Spanish adjective (mf) has to be translated as a feminine or a masculine adjective in Catalan. In that case, the symbol "GD" (for "gender to be determined") is used instead of the gender symbol. The word's gender will be determined by the structural transfer module, by means of a transfer rule (a rule that detects the gender of the preceding noun in this particular case).

Therefore, "GD" must be used only when translating from Spanish to Catalan, but not when translating from Catalan to Spanish, as in Spanish the gender will always be mf regardless of the gender of the original word.

In the bilingual dictionary you will need to add, in this case, more than one entry with direction indications, as you must specify in which translation direction the gender remains undetermined. The entries for this adjective should be as follows:

```
<e r="LR">
   <p>
     <l>canadiense<s n="adj"/><s n="mf"/></l>
     <r>canadenc<s n="adj"/><s n="GD"/></r>
   </p>
 </e>
 <e r="RL">
   <p>
     <l>canadiense<s n="adj"/><s n="mf"/></l>
     <r>canadenc<s n="adj"/><s n="f"/></r>
   </p>
 </e>
 <e r="RL">
   <p>
     <l>canadiense<s n="adj"/><s n="mf"/></l>
     <r>canadenc<s n="adj"/><s n="m"/></r>
   </p>
 </e>
```

"LR" means "left to right", and "RL", "right to left". Since Spanish is on the left and Catalan on the right, the adjective will be "GD" only when translating from Spanish to Catalan ("LR"). For the translation "RL" you need to create two entries, one for the adjective in feminine and another one for the adjective in masculine. The same principle applies when it is not possible to determine the number (singular or plural) of the target word for the same reasons mentioned above. For example, the Spanish noun "rascacielos" (skyscraper) is invariable in number, that is, it can be singular as well as plural (*un rascacielos, dos rascacielos*). In Catalan, on the other hand, the noun has a different inflection for the singular and for the plural (*un gratacel, dos gratacels*). In this case the symbol used is "ND" ("number to be determined") and the entries should be like this:

```
<e r="LR">
  <p>
    <l>rascacielos<s n="n"/><s n="m"/><s n="sp"/></l>
    <r>gratacel<s n="n"/><s n="m"/><s n="ND"/></r>
  </p>
</e>
<e r="RL">
  <p>
    <l>rascacielos<s n="n"/><s n="m"/><s n="sp"/></l>
    <r>gratacel<s n="n"/><s n="m"/><s n="pl"/></r>
  </p>
</e>
<e r="RL">
  <p>
    <l>rascacielos<s n="n"/><s n="m"/><s n="sp"/></l>
    <r>gratacel<s n="n"/><s n="m"/><s n="sg"/></r>
  </p>
</e>
```

For a more detailed description of this kind of entries, refer to section 3.1.2.4.2 of the system documentation.

Adding direction restrictions

In the previous example we have already seen the use of direction restrictions for entries with undetermined gender or number ("GD" or "ND"). Restrictions can also be used in other cases. It is important to note that the current version of Apertium can give only a single equivalent for each source-language lexical form (a lexical form is the lemma plus its grammatical information), that is, no word-sense disambiguation is performed. [Note: The system performs only part-of-speech disambiguation for homograph words, that is, for ambiguous words that can be analyzed as more than one lexical form, like vino in Spanish, that can mean both "wine" and "he/she came". This type of disambiguation is performed by the tagger.] When a lemma can be translated in two or more different ways, one has to be chosen (the most general, the most frequent, etc.). You can tell Apertium that a certain word has to be understood (analyzed) but not generated, as it is not the translation of any source-language lemma.

Let's see this with an example. The Spanish noun "muñeca" can be translated in two different ways in Catalan depending on its meaning: "canell" (wrist) or "nina" (doll). The context decides which translation is the correct one, but in its present state Apertium can not make such a decision (but see below the section on multiword units for ways to circumvent this problem). Therefore, you have to decide which word you want as an equivalent when translating from Spanish to Catalan. From Catalan to Spanish, both words can be translated as "muñeca" without any problem. You have to specify all these circumstances in the dictionary entries using direction restrictions ("LR" meaning "left to right", that is, Spanish-Catalan, and "RL" meaning "right to left", that is, Catalan-Spanish). If you decide to translate "muñeca" as "canell" in all cases, the entries in the bilingual dictionary shall be:

```
<e>
  <p>
    <l>muñeca<s n="n"/><s n="f"/></l>
    <r>canell<s n="n"/><s n="m"/></r>
  </p>
```

```
      </e>

      <e r="RL">
        <p>
          <l>muñeca<s n="n"/></l>
          <r>nina<s n="n"/></r>
        </p>
      </e>
```

This means that translation directions will be:

```
muñeca --> canell
muñeca <-- canell
muñeca <-- nina
```

```
(Note that that there is also a gender change in the case of "muñeca" (feminine) and "canell" (masculine))
```

It should be emphasized that a lemma can not have two translations in the target language, because the system would give an error when translating that lemma (see the section "Detecting errors" below to see how to find and correct these and other types of errors). When a word can be translated in two different ways in the target language in all contexts, you need to choose one as the translation equivalent and leave the other one as a lemma that can be analyzed but not generated, using direction restrictions like in the previous example. For example, the Catalan lemmas "mot" and "paraula" can be both translated into Spanish as "palabra" (word) and the entry in the bilingual dictionary should look like this:

```
      <e>
        <p>
          <l>palabra<s n="n"/></l>
          <r>paraula<s n="n"/></r>
        </p>
      </e>
      <e r="RL">
        <p>
          <l>palabra<s n="n"/><s n="f"/></l>
          <r>mot<s n="n"/><s n="m"/></r>
        </p>
      </e>
```

Therefore, for this lemmas the translation directions will be:

```
palabra --> paraula
palabra <-- paraula
palabra <-- mot
```

One may have to specify restrictions regarding translation direction also in monolingual dictionaries. For example, both Spanish forms cantaran or cantasen should be analyzed as cantar,verb, subjunctive imperfect, 3rd person plural, but when generating Spanish text, one has to decide which one will be generated. Monolingual dictionaries are read in two directions depending on its purpose: for the analysis, the reading direction is left to right; for the generation, right to left. Therefore, a word that must be analyzed but not generated must have the restriction "LR", and a word that must be generated but not analyzed must have the restriction "RL". The case of cantaran or cantasen must have already been taken care of in inflection paradigms and it is unlikely to be a problem for most people extending a dictionary. In some other cases it can be necessary to introduce a restriction in the word entries of monolingual dictionaries.

Adding multiwords

It is possible to create entries consisting of two or more words, if these words are considered to build a single "translation unit". These "multiword units" can also be useful when it comes to select the correct equivalent for a word inside a fixed expression. For example, the Spanish word "dirección" may be translated into two Catalan words: "direcció" (direction, management, directorate, steering, etc.) and "adreça" (address); including, for example, frequent multiword units such as "dirección general" --> "direcció general" (general directorate) and "dirección postal" --> "adreça postal" (postal address) may help get improved translations in some situations.

Multiword units can be classified basically into two categories: multiwords with inner inflection and multiwords without inner inflection.

Multiwords without inner inflection are just like the normal one-word entries, with the only difference that you need to insert the element (which represents a blank) between the individual words that make up the unit. Therefore, if you want to add, for example, the Spanish multiword "hoy en día" (nowadays), whose equivalent in Catalan is "avui dia", the entries you need to add to the different dictionaries are:

Spanish monolingual dictionary

```
<e lm="hoy en día">
  <i>hoy<b/>en<b/>día</i>
  <par n="ahora__adv"/>
 </e>
```

Catalan monolingual dictionary

```
<e lm="avui dia">
  <i>avui<b/>dia</i>
  <par n="ahir__adv"/>
</e>
```

Spanish-Catalan bilingual dictionary

```
<e>
  <p>
    <l>hoy<b/>en<b/>día<s n="adv"/></l>
    <r>avui<b/>dia<s n="adv"/></r>
  </p>
</e>
```

For Spanish-Galician pair, if you want to add, for example, the Spanish multiword "manga por hombro" (disarranged), whose equivalent in Galician is "sen xeito nin modo", the entries you need to add are:

Spanish monolingual dictionary

```
<e lm="manga por hombro">
    <i>manga<b/>por<b/>hombro</i>
    <par n="abajo__adv"/>
</e>
```

Galician monolingual dictionary

```
<e lm="sen xeito nin modo">
    <i>sen<b/>xeito<b/>nin<b/>modo</i>
    <par n="Deo_gratias__adv"/>
</e>
```

Spanish-Galician bilingual dictionary

```
<e>
   <p>
      <l>manga<b/>por<b/>hombro<s n="adv"/></l>
      <r>sen<b/>xeito<b/>nin<b/>modo<s n="adv"/></r>
   </p>
</e>
```

Multiwords with inner inflection consist of a word that can inflect (typically a verb) and an invariable element. For these entries you need to specify the inflection paradigm just after the word that inflects. The invariable part must be marked with the element <g> in the right side. The blanks between words are indicated, like in the previous case, with the element . Look at the following example for the Spanish multiword "echar de menos" (to miss), translated into Catalan as "trobar a faltar":

Spanish monolingual dictionary

```
<e lm="echar de menos">
   <i>ech</i>
      <par n="aspir/ar__vblex"/>
   <p>
      <l><b/>de<b/>menos</l>
      <r><g><b/>de<b/>menos</g></r>
   </p>
 </e>
```

Catalan monolingual dictionary

```
 <e lm="trobar a faltar">
   <i>trob</i>
      <par n="abander/ar__vblex"/>
   <p>
      <l><b/>a<b/>faltar</l>
      <r><g><b/>a<b/>faltar</g></r>
   </p>
 </e>
```

Spanish-Catalan bilingual dictionary

```
<e>
  <p>
    <l>echar<g><b/>de<b/>menos</g><s n="vblex"/></l>
    <r>trobar<g><b/>a<b/>faltar</g><s n="vblex"/></r>
 </p>
</e>
```

Note that the grammatical symbol is appended at the end, after the group marked with the <g>. It can be the case that a lemma is a multiword in one language and a single word in the other language. In that case, in the bilingual dictionary, the multiword will contain the <g> element and the single word will not. In the monolingual dictionaries, each entry will be created according to its type. Look at the following example for the Spanish multiword "darse cuenta" (to realize), translated into Catalan as the verb "adonar-se":

Spanish monolingual dictionary

```
<e lm="darse cuenta">
    <i>d</i>
       <par n="d/ar__vblex"/>
    <p>
       <l><b/>cuenta</l>
       <r><g><b/>cuenta</g></r>
    </p>
```

```
    </e>
```

Catalan monolingual dictionary

```
    <e lm="adonar-se">
      <i>adon</i>
      <par n="abander/ar__vblex"/>
    </e>
```

Spanish-Catalan bilingual dictionary

```
  <e>
    <p>
      <l>dar<g><b/>cuenta</g><s n="vblex"/></l>
      <r>adonar<s n="vblex"/></r>
    </p>
  </e>
```

Note that the enclitic pronouns (such as "-se") after both the Spanish and Catalan verb forms need not be specified and will be taken care of by structural transfer rules; the correct positioning of clitics is one of the main reasons for using the <g>... </g> labels around the invariable part of multi-word verbs.

The same principles and actions described for basic entries (gender and number change, direction restrictions, etc.) apply to all kinds of multiwords. For a more detailed description of multiword units, refer to section 3.1.2.5 of the system documentation.

Brief introduction to paradigms

The paradigms of the previous examples, as adverbs do not inflect, contain only the grammatical symbol of the lexical form, as you see in this example:

```
 <pardef n="ahora__adv">
      <e>
         <p>
            <l/>
            <r><s n="adv"/></r>
         </p>
      </e>
  </pardef>
```

Paradigms are build like a lexical entry. We have seen so far lexical entries where the common part of the lemma is put between :

```
<e lm="cósmico">
  <i>cósmic</i>
  <par n="absolut/o__adj"/>
</e>
```

But you can also express the same with a pair of strings: a left string <l> and a right string <r> inside a <p> element:

```
<e lm="cósmico">
   <p>
      <l>cósmic</l>
      <r>cósmic</r>
   </p>
   <par n="absolut/o__adj"/>
</e>
```

These two entries are equivalent. The use of the <i> element helps get more simple and compact entries, and you can use it when the left side and the right side of the string pair are identical. As has been explained before, monolingual dictionaries are read LR for the analysis of a text and RL for the generation. Therefore, when there is some difference between the analysed string and the generated string (not very usual) the entry can not be written using the <i> element. In paradigms, the left and right strings are never identical, since the right side must contain the grammatical symbols that will go through all the modules of the system.

Adding structural transfer (grammar) rules

Structural transfer rules carry out transformations to the disambiguated text, which are needed because of grammatical, syntactical and lexical divergences between the two languages involved (gender and number changes to ensure agreement in the target language, word reorderings, changes in prepositions, etc.). The rules detect patterns (sequences) of source text lexical forms and apply to them the corresponding transformations. The module detects the patterns in a left-to-right, longest-match way; for example, the phrase "the big cat" will be detected and processed by the rule for determiner - adjective - noun and not by the rule for determiner - adjective, since the first pattern is longer. If two patterns have the same length, the rule that applies is the one defined in the first place.

The structural transfer module (generated from the structural transfer rules file) calls the lexical transfer module (generated from the bilingual dictionary) all through the process to determine the target language equivalents of the source language lexical forms.

The structural transfer rules are contained in an XML file, one for each translation direction (for example, for the translation from Spanish to Catalan, the file is apertium-es-ca.trules-es-ca.xml). You need to edit this file if you want to add or change transfer rules.

Rules have a pattern and an action part. The pattern specifies which sequences of lexical forms have to be detected and processed. The action describes the verifications and transformations that need to be done on its constituents. Usual transformation operations (such as gender and number agreement) are defined inside a macroinstruction which is called inside the rule. At the end of the action part of the rule, the resulting lexical forms in the target language are sent out so that they are processed by the next modules in the translation system.

A transfer rules file contains four sections with definitions of elements used in the rules, and a fifth section where the actual rules are defined. The sections are the following:

- <section-def-cats>: This section contains the definition of the categories which are to be used in the rule patterns (that is, the type of lexical forms that will be detected by a certain rule). For the rule presented below, the categories "det" and "nom" (determiner and noun) need to be defined here. Categories are defined specifying the grammatical symbols that the lexical forms have. An asterisk indicates that one or more grammatical symbols follow the ones specified. The following is the definition of the category "det", which groups determiners and predeterminers in the same category since they play the same role for transfer purposes:

```
<def-cat n="det">
        <cat-item tags="det.*"/>
        <cat-item tags="predet.*"/>
</def-cat>
```

It is also possible to define as a category a certain lemma, like the following for the preposition "en":

```
<def-cat n="en">
        <cat-item lemma="en" tags="pr"/>
</def-cat>
```

- <section-def-attrs>: This section contains the definition of the attributes that will be used inside of the rules, in the action part. You need attributes for all the categories defined in the previous section, if they are to be used in the action part of the rule (to make verifications on them or to send them out at the end of the rule), as well as for other attributes needed in the rule (such as gender or number). Attributes have to be defined using their corresponding grammatical symbols and can not have asterisks; its name must be unique. The following are the definitions for the attributes "a_det" (for determiners) and "gen" (gender):

```
<def-attr n="a_det">
        <attr-item tags="det.def"/>
        <attr-item tags="det.ind"/>
        <attr-item tags="det.dem"/>
        <attr-item tags="det.pos"/>
        <attr-item tags="predet"/>
</def-attr>

<def-attr n="gen">
        <attr-item tags="m"/>
        <attr-item tags="f"/>
        <attr-item tags="mf"/>
        <attr-item tags="nt"/>
        <attr-item tags="GD"/>
</def-attr>
```

- <section-def-vars> : This section contains the definition of the variables used in the rules.

```
<def-var n="interrogativa"/>
```

- <section-def-macros> : Here the macroinstructions are defined, which contain sequences of code that are frequently used in the rules; this way, linguists do not need to write the same actions repeatedly. There are, for example, macroinstructions for gender and number agreement operations.

- <section-rules> : This is the section where the structural transfer rules are written.

The following is an example of a rule which detects the sequence determiner -- noun:

```
<rule>
      <pattern>
          <pattern-item n="det"/>
          <pattern-item n="nom"/>
      </pattern>
      <action>
          <call-macro n="f_concord2">
              <with-param pos="2"/>
              <with-param pos="1"/>
          </call-macro>
          <out>
              <lu>
                  <clip pos="1" side="tl" part="whole"/>
              </lu>
              <b pos="1"/>
              <lu>
                  <clip pos="2" side="tl" part="whole"/>
              </lu>
          </out>
     </action>
</rule>
```

Part of the action performed on this pattern is specified inside the macroinstruction "f_concord2", which is defined in the <section-def-macros>. It performs gender and number agreement operations: if there is a gender or number change between the source language and the target language (in the noun), the determiner changes its gender or number accordingly; furthermore, if gender or number are undetermined ("GD" or "ND", as explained in the previous section "Adding words to monolingual and bilingual dictionaries"), the noun receives the correct gender or number values from the preceding determiner. In the Apertium es-ca, es-gl and es-pt systems, there are agreement macroinstructions defined for one, two, three or four lexical units (f_concord1, f_concord2, f_concord3, f_concord4). When calling the macroinstructions in a rule, it must be specified which is the main lexical unit (the one which most heavily determines the gender or number of the other lexical units) and which other lexical units of the pattern have to be included in the agreement operations, in order of importance. This is done with the <with-param pos=""/> element. In the previous rule, the main lexical unit is the noun (position "2" in the pattern) and the second one is the determiner (position "1" in the pattern).

After the pertinent actions, the resulting lexical forms are sent out, inside the <out> element. Each lexical unit is defined with a <clip>. Its attributes mean the following:

- pos: refers to the position of the lexical form in the pattern. "1" is the first lexical form (the determiner) and "2" the second one (the noun).
- side: indicates if the lexical form is in the source language ("sl") or in the target language ("tl"). Of course, words are sent out always in the target language; source language lexical forms may be needed inside of a rule, when testing its attributes or characteristics.

- part: indicates which part of the lexical form is referred to in the 'clip'. You can use some predefined values:
  - `whole`: the whole lexical form (lemma and grammatical symbols). Used only when sending out the lexical unit (inside an <out> element).
  - `lem`: the lemma of the lexical unit
  - `lemh`: the head of the lemma of a multiword with inner inflection
  - `lemq`: the queue of a lemma of a multiword with inner inflection

using lemh and lemq for multiwords

As well as these predefined values, you can use any of the attributes defined in <section-def-attrs> (for example "gen" or "a_det").

The values "lemh" and "lemq" are used when sending out multiwords with inner inflection (see the section "Adding multiwords") in order to place the head and the queue of the lemma in the right position, since the previous module moved the queue just after the lemma head for various reasons. In pratice, in our system, this means that you must use these values instead of 'whole' when sending out verbs, since in our dictionaries multiwords with inner inflection are always verbs and, if you use the value "whole" when sending them out, the multiword would not be well formed (the head and the queue of the lemma would not have the correct position).

Therefore, a rule that has a verb in its pattern must send the lexical forms like in the following two examples:

```
<rule>
       <pattern>
           <pattern-item n="verb"/>
       </pattern>
       <action>
           <out>
              <lu>
                  <clip pos="1" side="tl" part="lemh"/>
                  <clip pos="1" side="tl" part="a_verb"/>
                  <clip pos="1" side="tl" part="temps"/>
                  <clip pos="1" side="tl" part="persona"/>
                  <clip pos="1" side="tl" part="gen"/>
                  <clip pos="1" side="tl" part="nbr"/>
                  <clip pos="1" side="tl" part="lemq"/>
              </lu>
          </out>
       </action>
</rule>


<rule>
      <pattern>
         <pattern-item n="verb"/>
         <pattern-item n="prnenc"/>
     </pattern>
     <action>
         <out>
             <mlu>
                <lu>
                   <clip pos="1" side="tl" part="lemh"/>
                   <clip pos="1" side="tl" part="a_verb"/>
                   <clip pos="1" side="tl" part="temps"/>
                   <clip pos="1" side="tl" part="persona"/>
```

```
                    <clip pos="1" side="tl" part="nbr"/>
                </lu>
                <lu>
                    <clip pos="2" side="tl" part="lem"/>
                    <clip pos="2" side="tl" part="a_prnenc"/>
                    <clip pos="2" side="tl" part="persona"/>
                    <clip pos="2" side="tl" part="gen"/>
                    <clip pos="2" side="tl" part="nbr"/>
                    <clip pos="1" side="tl" part="lemq"/>
                </lu>
            </mlu>
        </out>
    </action>
</rule>
```

The first rule detects a verb and places the queue in the correct place, after all the grammatical symbols. The lexical unit is sent specifying the attributes separately: lemma head, lexical category (verb), tense, person, gender (for the participles), number and lemma queue.

The second rule detects a verb followed by an enclitic pronoun and sends the two lexical forms specifying also the attributes separately; the first lexical unit consists of: lemma head, lexical category (verb), tense, person and number; the second lexical unit consists of: lemma, lexical category (enclitic pronoun), person, gender, number and lemma queue. This way, the queue of the lemma is placed after the enclitic pronoun. The two lexical units (verb and enclitic pronoun) are sent inside a <mlu> element, since they have to reach the morphological generator as a multilexical unit (multiword).

If you want to add a new transfer rule, you have to follow these steps:

1. Specify which pattern you want to detect. Bear in mind that words are processed only once by a rule, and that rules are applied left to right and choosing the longest match. For example, imagine you have in your transfer rules file only two rules, one for the pattern "determiner - noun" and one for the pattern "noun - adjective". The Spanish phrase "el valle verde" (the green valley)" would be detected and processed by the first one, not by the second. You will need to add a rule for the pattern "determiner - noun - adjective" if you wish that the three lexical units are processed in the same pattern.
2. Describe the operations you want to perform on the pattern. In the Apertium es-ca, es-gl and es-pt, simple agreement operations (gender and number agreement) are easy to perform in a rule by means of a macroinstruction. To perform other operations, you will need to use more complicated elements; for a more detailed description of the language used to create rules, refer to the section 3.4.2 of the system documentation. You can also read the DTD of the structural transfer rules file (transfer.dtd) provided with the Apertium package, in which all the elements of the language are described in English.
3. Send the lexical units of the pattern in the target language inside an <out> element. Each lexical unit must be included in a <lu> element. If two or more lexical units must be generated as a multilexical unit (only for enclitic pronouns in the present language pairs), they must be grouped inside a <mlu> element.

All the words that are detected by a rule (that are part of a pattern) must be sent out at the end of the rule so that the next module (the generator) receives them. If a lexical unit is detected by a pattern and is not included in the <out> element, it will not be generated.

Adding data to the part-of-speech tagger

To be written.

Retraining the part-of-speech tagger

Main article: Tagger training

To be written.

Detecting errors

It is easy to make errors when adding new words or transfer rules to the Apertium system.

It is possible that, when compiling the new files, the system displays an error message. In this case, this is a formal error (a missing XML tag, a tag that is not allowed in a certain context, etc.). You just have to go to the line number indicated by the error message, correct the error and compile again. Other types of errors are not detected when compiling, but can make the system mistranslate a word or give an incomprehensible text string. These are linguistic errors, which can be detected and corrected with the tips given in this chapter. The following information is for Linux users, since Apertium works for the moment only with this OS.

As all the data processed by the system, from the original text to the translated text, circulate between the eight modules of the system in text format, it is possible to stop the text stream at any point to know what is the input or the output of a certain module. You just have to type the right commands in the terminal using a pipeline structure so that the standard output of one is used as the standard input for the next one.

Using the echo or cat commands, you can send a text through one or more modules to analyse their output and detect the origin of the error. You have to move to the directory where the linguistic data are saved and type the commands explained below.

The morphological analyser output

To know how a word is analysed by the translator, type the following in the terminal (example for the Catalan word gener):

```
$ echo "gener" | apertium-destxt | lt-proc ca-es.automorf.bin
```

You can replace ca-es with the translation direction you want to test.

The output in Apertium should be:

```
^gener/gener<n><m><sg>$^./.<sent>$[][]
```

The string structure is: `^word/lemma<morphological analysis>$`. The `<sent>` tag is the analysis of the full stop, as every sentence end is represented as a full stop by the system, whether or not explicitly indicated in the sentence.

The analysis of an unknown word is (ignoring the full stop information):

```
^genoma/*genoma$
```

and the analysis of an ambiguous word:

```
^casa/casa<n><f><sg>/casar<vblex><pri><p3><sg>/casar<vblex><imp><p2><sg>$
```

Each lexical form (lemma plus morphological analysis) is presented as a possible analysis of the word casa.

The tagger output

To know the output of the tagger for a source language text, type the following in the terminal (example for the Catalan-Spanish direction):

```
$ echo "gener" | apertium-destxt | lt-proc ca-es.automorf.bin | apertium-tagger --tagger ca-es.prob
```

The output will be:

```
^gener<n><m><sg>$^.<sent>$[][]
```

The output for an ambiguous word will be like the one above, since the tagger chooses one lexical form. Therefore, the output for casa in Catalan will be, for example (depending on the context):

```
^casa<n><f><sg>$^.<sent>$[][]
```

The pretransfer output

This module applies some changes to multiwords (move the lemma queue of a multiword with inner inflection just after the lemma head). To know its output, type:

```
$ echo "gener" | apertium-destxt | lt-proc ca-es.automorf.bin | apertium-tagger -g ca-es.prob | apertium-pretransfer
```

Since gener is not a multiword in the dictionaries, this module does not alter its input.

The structural transfer output

To know how a word, phrase or sentence is translated into the target language and processed by structural transfer rules, type the following in the terminal:

```
$ echo "gener" | apertium-destxt | lt-proc ca-es.automorf.bin | apertium-tagger -g ca-es.prob | apertium-pretransfer \
| apertium-transfer trules-ca-es.xml trules-ca-es.bin ca-es.autobil.bin
```

The output for the Catalan word gener will be:

```
^enero<n><m><sg>$
```

Analysing how a word or phrase is output by this module can help you detect errors in the bilingual dictionary or in the structural transfer rules. Typical bilingual dictionary errors are: two equivalents for the same source language lexical form, or wrong assignment of morphological symbols. Errors due to structurual transfer rules vary a lot depending on the actions performed by the rules.

The morphological generator output

To know how a word is generated by the system, type the following in the terminal:

```
$ echo "gener" | apertium-destxt | lt-proc ca-es.automorf.bin | apertium-tagger -g ca-es.prob | apertium-pretransfer \
| apertium-transfer trules-ca-es.xml trules-ca-es.bin ca-es.autobil.bin | lt-proc -g ca-es.autogen.bin
```

With this command you can detect generation errors due to an incorrect entry in the target language monolingual dictionary or to a divergence between the output of the bilingual dictionary (the output of the previous module) and the entry in the monolingual dictionary.

The correct output for the input "gener" would be:

```
enero.[][]
```

There are in this case no morphological symbols, and the word appears inflected.

The post-generator output

It is not very usual to have errors due to the postgenerator, because of its generally small size and the fact that it is seldom changed after adding usual combinations, but you can also test how a source language text comes out of this module, by typing:

```
$ echo "gener" | apertium-destxt | lt-proc ca-es.automorf.bin | apertium-tagger -g ca-es.prob | apertium-pretransfer \
| apertium-transfer trules-ca-es.xml trules-ca-es.bin ca-es.autobil.bin | lt-proc -g ca-es.autogen.bin | lt-proc -p ca-es.autopgen.bin
```

The final output

You can put all the modules of the system in the pipeline structure and see how a source language text goes through all the modules and gets translated into the target language. You just have to add the reformatter to the previous command:

```
$ echo "gener" | apertium-destxt | lt-proc ca-es.automorf.bin | apertium-tagger -g ca-es.prob | apertium-pretransfer \
| apertium-transfer trules-ca-es.xml trules-ca-es.bin ca-es.autobil.bin | lt-proc -g ca-es.autogen.bin | lt-proc -p ca-es.autopgen.bin | apertium-retxt
```

This is the same as using the "apertium" shell script provided by the Apertium package:

```
$ echo "gener" | apertium -d . ca-es
```

(The full stop indicates the directory where the linguistic data are saved.)

Of course, instead of typing all the presented commands every time you need to test a translation, you can create shell scripts for every action and use them to test the output of each module.

Error examples

To be written.

Generating a new Apertium system from modified data

If you make changes to any of the linguistic data files of Apertium (the dictionaries, the transfer rules, the tagger file or corpora or the postgeneration dictionaries), the changes will not be applied until you recompile the modules. To do this, type 'make' in the directory where the linguistic data are saved (apertium-es-ca, apertium-es-gl or what may be applicable) so that the system generates the new binary files.

Note that if a pair depends on monolingual data, you may have to first type *make* in the monolingual directory, and then in the pair directory, to test the pair.

If changes were made to the tagger file or to the corpora used to train the tagger, you will need also to retrain the tagger. In the same directory mentioned above, after typing make, type make train in order to perform the retraining.

Now the Apertium translator will work using the recently added data.

See also

- Comment contribuer à une paire de langues existante
- Building dictionaries
- Finding_errors_in_dictionaries
- A_long_introduction_to_transfer_rules
- Apertium New Language Pair HOWTO

Source

The original version was transferred from source http://apertium.sourceforge.net/extending.html.