Chunking

En français

Shallow transfer

Shallow transfer means there is no parse trees (which are used in "deep transfer"). But then how is reordering of the phrase then going to happen?

By chunking (in three-stages): First we reorder words in the chunk, then we reorder chunks.

- first, we match phrase patterns, like adj+noun or adj+adj+noun
- from these, we make a 'pseudo lemma', with a tag containing the type - normally 'SN' (noun phrase) or SV (verb phrase)
- then, we translate based on these pseudo words breaking the language down to its bare essentials, basically

Contents

Chunking explained

Our rules are based on the source language patterns; we need to use chunking for f.ex. English-Esperanto, so the first task is to identify those patterns.

```
the man sees the girl
```

Chunking:

```
SN(the man) SV(sees) SN(the girl)
```

(Normally, in English those are 'NP' and 'VP' for 'noun phrase' and 'verb phrase' respectively, but we'll stick to the established convention in apertium)

Two rules are needed to make those chunks: further chunking rules can match 'the tall man' 'my favourite Spanish friend' 'the prettiest Polish girl' etc. as SN; 'was going', 'had been going', 'must have been going' as SV. We first consider these patterns separately, but tag the chunks with whatever information will be useful later.

So the chunks are normally given a 'pseudo lemma' that matches the pattern that matched them ('the man', 'my friend' will be put in a chunk called 'det_nom', etc.), the first tag added is the phrase type; after that, tags that are needed in the next set of rules. Essentially, we're treating phrase chunks in the same way that the morphological analyser treats lexemes ('surface forms').

So, taking 'big cat', we would get:

```
^adj_nom<SN><sg><CD>{^granda<ad><2><3>$ ^kato<n><2><3>$}$
```

The numbers in the lemma tags (here <2><3>) mean 'take the information from chunk tag number #'. CD means 'Case to be Determined (it's not fully established, as GD and ND are, but it's the logical one to use).

So, with a simple SN SV SN, we can have a rule that outputs the same things in the same order, but changes the 'CD' of SN number 1 to 'nom', and of SN number 2 to 'acc'.

Example

```
I saw a signal
```

becomes after tagger disambiguation

```
^prpers<prn><subj><p1><mf><sg>$
^see<vblex><past>$
^a<det><ind><sg>$
^signal<n><sg>$.
```

which is transfered and chunked into

```
^prnpers<SN><p1><mf><sg>{^prpers<prn><subj><2><3><4>$}$
^verb<SV><past>{^vidi<vblex><past>$}$
^nom<SN><sg><nom>{^signalo<n><2><3><4>$}$.
```

and transformed by rule SN SV SN<nom> -> SN SV SN<acc>

```
^prnpers<SN><p1><mf><sg>{^prpers<prn><subj><2><3><4>$}$
^verb<SV><past>{^vidi<vblex><past>$}$
^nom<SN><sg><acc>{^signalo<n><2><3><4>$}$.
```

Note how the chunk has now tags nom<SN><sg><acc> and therefore ^signalo<n><2><3><4>$ gets these tags when unchunking:

```
^prpers<prn><subj><p1><mf><sg>$
^vidi<vblex><past>$
```

```
^signalo<n><sg><acc>$.
```

See also

- Chunking: A full example
- Apertium stream format#Chunks
- Preparing to use apertium-transfer-tools
- English and Esperanto

External links

- wikipedia (http://en.wikipedia.org/wiki/Chunking_(computational_linguistics))
- Chunking (http://nltk.googlecode.com/svn/trunk/doc/book/ch07.html) (Natural Language Toolkit)
- CRFChunker (http://crfchunker.sourceforge.net/) (Conditional Random Fields English Phrase Chunker)
- JTextPro (http://jtextpro.sourceforge.net/) (A Java-based Text Processing Toolkit)