# A first step towards interactivity and language tools convergence

Arnaud Vié[1] Luis Villarejo Muñoz[2]
Mireia Farrús Cabeceran[2] Jimmy O'Regan[3]

[1]Informations Systems Engineering
Grenoble INP - Ensimag
arnaud.vie@ensimag.fr

[2]Learning Technologies Office
Universitat Oberta de Catalunya
lvillarejo@uoc.edu,
mfarrusc@uoc.edu

[3]Eolaistriu Technologies
joregan@gmail.com

# Two Main Uses of MT

**Assimilation**

understanding the text

**Dissemination**

producing text for consumption

# Dissemination

No MT system is perfect.
Post-editing is required to produce text fit for human consumption.

# Post-Edition

MT Post-Edition is an active research area, with two main aims:

- Improve the translation.
- Improve the MT system itself.

# Apertium and Post-Editing

Apertium currently has *no* post-editing facilities.
This is an obvious disadvantage to the user, who must use other means to improve the translation.
A less obvious disadvantage is to the developers of Apertium language pairs, who miss out on user feedback that could be used to improve the system.

# Two Types of Edits

**Brief Edits**

small, local changes (mostly syntactic)

**Full Edits**

larger changes (mostly stylistic)

# Full Edits

The most famous example of a Full Edit system is Google Translate's "Suggest a Better Translation" feature.
Users click on a sentence, and type a replacement. Google (after checking for a number of undesirables) later add this to their corpora for later integration.

# Full Edits

The Open Source Computer Assisted Translation tools Virtaal and OmegaT both include the facility to have the current sentence translated using (mostly) online MT services.
To a certain degree, they can be considered Open Source MT post-edition tools, which allow Full Edits.
However, using a desktop system with online translations brings the disadvantages of desktop software to an online service (and vice versa).

# Brief Edits

Ironically, Google Translate is also the most famous example of a post-edition interface including Brief Edits.
The "Suggest" feature has recently been replaced with a simpler feature that allows the user to click on an individual word, and choose an alternative translation.

# Primary Goals

- Fully online
- Full Edits
- Easy to integrate with existing installations
- Extensible
- Feedback
- Integration with existing tools

# Fully online

Most Apertium users use it via a web interface.

# Full edits

The translator should be in control, not the interface

# Integration with Existing Installations

Most current web interfaces to Apertium are written in PHP; this interface was also written in PHP to make integration easier.

# Extensible

The interface was designed to be a *framework* for post-edition.
Lacking current work in Apertium on automated post-edition, we cannot know what future needs will be.
With the rapid pace of language pair development, we cannot know what future language pairs will need from post-edition.
We don't assume.

# Feedback

The interface should log user changes, for protential use in improving language pairs, and in developing automated post-edition tools.

# Integration with existing tools

We should leverage other Open Source projects where appropriate.
Spell checking is provided by Aspell.
Grammar checking is provided by LanguageTool.

# Translation Memory

The most obvious way to reuse edits is by using Translation Memory.
Apertium already includes support for TMX.
Does not work particularly well below the sentence level.
The mALIGNa sentence alignment tool is integrated, to allow creation of TMX files from edited sentences.

# Edit Logging

Realtime keystroke logging.
Uses browser events.
Associated events are aggregated and "promoted": a deleted word is logged as a deleted word, not as a series of character deletions.

# Word Objects

Each word is represented by an object containing:

- Current text (mutable)
- Original text (immutable)
- Reference to containing sentence object
- Reference to containing document node
- Position within the document node
- Reference to previous and next word objects

# Sentence Objects

Each sentence is represented by an object containing:

- References to its first and last words
- References to previous and next sentence objects

# Logging

Logging is invoked whenever editing occurs.
The logging module aggregates character level events and promotes them to word level events (deletion, insertion) or sentence level events (joining/splitting sentences) as appropriate.

# Logging

This aggregation is flexible: it may be modified to aggregate events in different ways, to provide more user feedback, etc., without changing the character-level logging system, which performs the bulk of the work.

# Formatted Documents

Apertium operates as a series of programs in a Unix-like pipeline: each program reads from standard input, and writes to standard output.
The first and last programs of a typical pipeline take care of removing formatting information, and restoring it, respectively.
This formatting information is retained inline, escaped as *superblanks*: extended text contained within braces, which are treated as spaces by the translation components.

# Superblanks

Similarly to the convention used in Translation Memory software, superblanks are presented to the user as a set of colour-coded (greyed), immutable strings in the interface, so the user can know at all times where to move words in relation to formatting.

# Spell checking and Grammar checking

All checking is performed server-side, for consistency, and to reduce administrative overhead.
When the user presses the "Check for Mistakes" button, spelling and grammar errors in the text are underlined (in red and blue, respectively). The user may then click on a word to see any suggestions the server may have to offer.

# External dictionaries

Mistake suggestions may be accompanied by translations of the suggestions provided by external web-based dictionaries, to help the user to find the intended meaning.
Dictionaries are provided using OpenSearch XML files (used by the search feature in the Firefox and Chrome browsers), for which a large number of dictionaries have ready made configurations.

# Search and Replace

The interface includes a search and replace feature.
In addition to the usual 'case sensitive'/'case insensitive' features, the interface includes an 'Apply source case' option, which takes case information from the original string and applies it to the replacement string.

# Future/Ongoing Work

The current interface has mostly been tested in Firefox: more testing is required for other browsers.
More work is required to improve the response in the interface.
TMX use with Apertium' built-in support is not currently optimal; OmegaT recently added an option to act as an automatic sentence translation system. We are investigating integrating it.
We are also looking into allowing After the Deadline to be used as an alternative to LanguageTool.

# Thanks

Questions?

# But first...

Arnaud could not join us today, because of his studies.
I hope you will join me in wishing him luck!

# Copying

This presentation is Open Content