CASMACAT: An Open Source Workbench for Advanced Computer Aided Translation

Vicent Alabau[a], Ragnar Bonk[b], Christian Buck[c], Michael Carl[b],
Francisco Casacuberta[a], Mercedes García-Martínez[b], Jesús González[a],
Philipp Koehn[c], Luis Leiva[a], Bartolomé Mesa-Lao[b], Daniel Ortiz[a],
Herve Saint-Amand[c], Germán Sanchis[a], Chara Tsoukala[c]

[a] Institut Tecnològic d'Informàtica, Universitat Politècnica de València, Spain
[b] Copenhagen Business School, Department of International Business Communication, Denmark
[c] School of Informatics, University of Edinburgh, Scotland

Abstract

We describe an open source workbench that offers advanced computer aided translation (CAT) functionality: post-editing machine translation (MT), interactive translation prediction (ITP), visualization of word alignment, extensive logging with replay mode, integration with eye trackers and e-pen.

1. Introduction

The use of machine translation technology among professional human translators is taking hold rapidly, but there is only very limited research on this man-machine collaboration, especially compared to the vast current research push on core machine translation technology. We believe that big part of this reason is that there are no sufficient open source platforms that lower the barrier to entry.

To resolve this, two EU-funded research projects, CASMACAT[1] and MATECAT,[2] are committed to develop an open source workbench targeted both at researchers to investigate novel and enhanced types of assistance and at professional translators for actual use. Through this combined effort, we hope to kick-start broader research into computer aided translation methods, facilitating diverse translation process studies, and reach volunteer and professional translators without advanced technical skills. At the mid-point of the 3-year projects, we release this tool as open source software. In this paper, we focus on CASMACAT's contributions and give instructions for installation and use of the workbench.

2. Related Work

A number of academic studies have shown that post-editing machine translation can be more efficient than translation from scratch (Plitt and Masselot, 2010; Skadiņš et al., 2011; Pouliquen et al., 2011; Federico et al., 2012), as is also evident from recent trends in industry adoption. But post-editing machine translation is not the only approach. The idea of so-called interactive machine translation was pioneered by the TRANSTYPE project (Langlais et al., 2000) and has been further developed in the following decade (Barrachina et al., 2009; Koehn, 2010).

We are not aware of any fully featured open source tool for computer aided translation research. A related open source project is OMEGAT, an editor with translation memory system, written in Java, targeted at freelance translators. We will explore integration of the functionalities of the CASMACAT workbench into this tool in the future.

3. Usage

The CASMACAT UI consists of views designated for different tasks. The translate view is its central view, where the user can translate a document and post-editing assistance and logging takes place. Other views offer a way to upload new documents or to manage the documents that are already in the system. Also, a replay mode has been implemented. The different views will now be shown and described in the sequence they are typically used.

3.1. Upload

If the user opens the default URL without giving any special parameters, she is taken to the upload view. This is currently the entry point of the application. At this point a user can specify one or several documents to upload and to translate. The documents uploaded must be in XLIFF format. The language pair can either be chosen

---

[1] http://www.casmacat.eu/
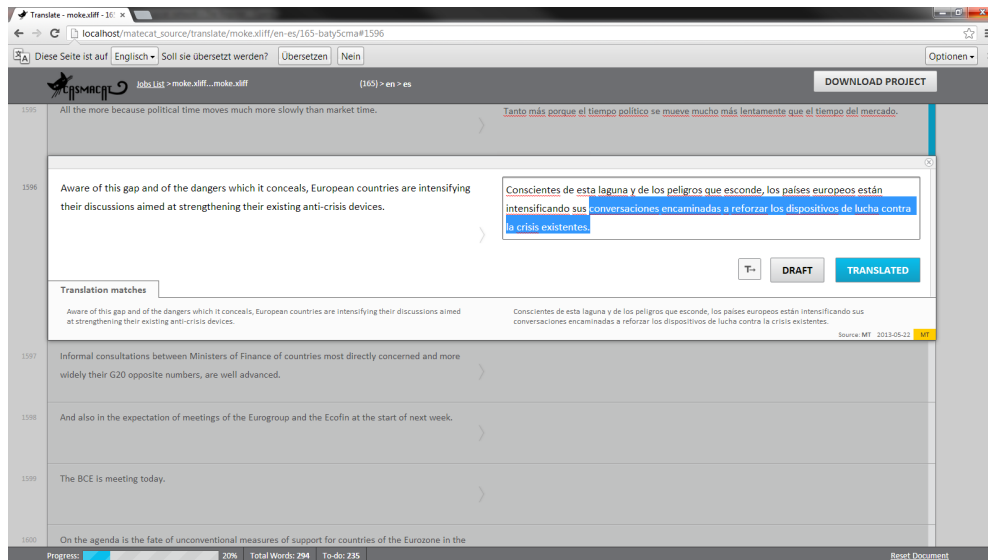
[2] http://www.matecat.com/

Figure 1. Translate view with post-editing configuration

manually or auto-detected from the XLIFF file. If several documents are uploaded at once, they are bundled into one job and are translated in a sequence. If the user clicks on the *Start Translating* button she is taken to the translate view and can start working.

3.2. Editing

In the translate view, the user can now translate the document (see Figure 1). The document is presented in segments, while the currently active segment is highlighted and assistance is provided for this segment. If using the post-editing configuration without ITP up to three MT or TM suggestions are provided, from which the user can choose. The user can use shortcuts, for instance, to go to the next segment or to copy the source text to the target. The user can assign different states to a segment, for instance, *translated* for finished ones or *draft* for segments, where she is not yet sure about the translation and she wants to review later. When finished, the *Download Project* button may be used to download the translated document, again in the XLIFF format.
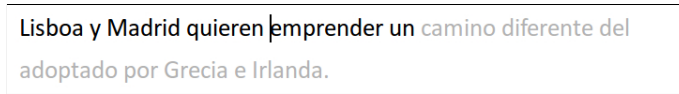
> Lisboa y Madrid quieren emprender un camino diferente del
> adoptado por Grecia e Irlanda.

Figure 2. Interactive Translation Prediction

4. Features

In this section we present a short description of the main advanced CAT features that we implemented in the workbench. The common goal of these features is to boost translator productivity.

4.1. Post-Editing Machine Translation

The default mode of the workbench is post-editing of either machine translation output or of matches from translation memory systems. This mode of operation is the minimal deviation from traditional work practice of professional translators, and hence the most conservative type of assistance offered.

4.2. Intelligent Autocompletion

The main alternative is interactive translation prediction, where new machine translation predictions are generated every time a keystroke is detected by the system (Barrachina et al., 2009). In such event, the system produces a prediction for the rest of the sentence according to the text that the user has already entered. This prediction is placed at the right of the text cursor.

Providing the user with a new prediction whenever a key is pressed has been proved to be cognitively demanding (Alabau et al., 2012). Therefore, we decided to limit the number of predicted words that are shown to the user by only predicting up to the first erroneous word according to confidence measures.

In our implementation, pressing the Tab key allows the user to ask the system for the next set of predicted words. See Figure 2 for a screenshot.

4.3. Confidence Measures

Confidence measures inform the user about which part of the translation is more likely to be wrong than others (González-Rubio et al., 2010). We use confidence measures under two different criteria. On the one hand, we highlight in red color those translated words that are likely to be incorrect. We use a threshold that favors precision in detecting incorrect words. On the other hand, we highlight in orange color those translated words that are dubious for the system. In this case, we use a threshold that favors recall.

4.4. Search and Replace

Most of the computer-assisted translation tools provide the user with intelligent search and replace functions for fast text revision. Our workbench features a straightforward function to run search and replacement rules on the fly. Whenever a new replacement rule is created, it is automatically populated to the forthcoming predictions made by the system, so that the user only needs to specify them once.

4.5. Word Alignment Information

Alignment of source and target words is an important part of the translation process (Brown et al., 1993). In order to display the correspondences between both the source and target words, this feature was implemented in a way that every time the user places the mouse (yellow) or the text cursor (cyan) on a word, the alignments made by the system are highlighted.

4.6. E-Pen Interaction

E-pen interaction should be regarded as a complementary input rather than a complete replacement of the keyboard. The user can interact with the system by writing on a special area of the user interface. We decided to use MINGESTURES (Leiva et al., 2013), a highly accurate, high-performance gestures for interactive text editing.

Although in principle it would be interesting to allow the user to introduce arbitrary strings and gestures, in this approach we have decided to focus on usability. We believe that a fast response and a good accuracy are critical for user acceptance.

4.7. Logging and Replay

The workbench implements detailed logging of user activity, which enables both automatic analysis of translator behavior by aggregating statistics and enabling replay of a user session. Replay takes place in the translate view of the UI, it shows the screen at any time exactly the way the user encountered it when she interacted with the tool.

4.8. Eye-Tracking

One of the core goals of the CASMACAT project is the study of translator behavior. To better observe the activities of translators, we use eye tracking. This allows us to detect and record the exact screen position of the current focus of attention. Alongside the other logged information such as key logging, enables *translation process study*., i.e., the analysis of the behavior of the translator, opposed to just *translation product study*, i.e., the analysis of the final translation.

Eye tracking is integrated into the CASMACAT workbench using a special plugin for the browser. With this plugin, the eye tracking information is accessible to the

Javascript   PHP

GUI ⇄ HTTP web
              server

                                        MT
                                      server
                    HTTP

web socket

              CAT
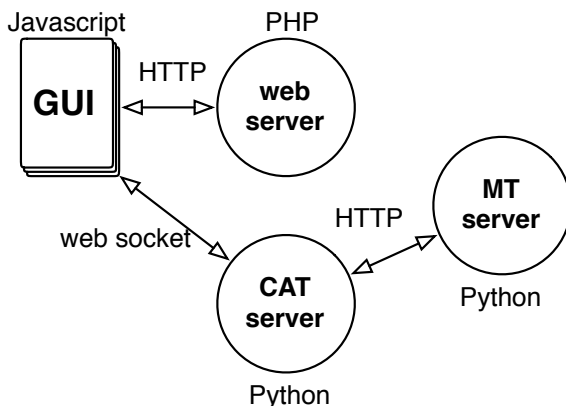   server   Python

              Python

*Figure 3. Modular design of the workbench: Web-based components (GUI and web
server), CAT server and MT server are independent and can be swapped out*

Javascript GUI and can be sent to the web server to be stored for later analysis. The
eye tracking information is also visualized in the replay mode.

5. Implementation

The tool is developed as a web-based platform using HTML5 and Javascript in the
Browser and PHP in the backend, supported by a CAT and MT server that run as
independent process (both implemented in Python but integrating tools written in
various other programming languages).

The overall design of the CASMACAT workbench is very modular. There are three
independent components (see also Figure 3): the GUI/web server, the CAT server
and the MT server. We modularize these components by clearly specified API calls,
so that alternative implementations can be used as well.

5.1. Computer Aided Translation Server

The computer aided translation (CAT) server is implemented in Python with the
Tornadio library. It uses *socket.io* to keep a web socket connection with the Javascript
GUI. Keep in mind that especially interactive translation prediction requires very
quick responses from the server. Establishing an HTTP connection through an Ajax
call every time the user presses a key would cause significant overhead.

A typical session with interactive translation prediction takes place as follows:
- The user moves to a new segment in the GUI.
- The GUI sends a **startSession** request to the CAT tool, passing along the input
  sentence.
- The GUI and CAT server establish a web socket connection.

- The CAT server requests and receives from the MT server the sentence translation and the search graph.
- The CAT server sends back the translation to the GUI and keeps the search graph in memory.
- The user starts typing (approving some of the translation or making corrections).
- At each key stroke, the GUI sends a request to the CAT server, for instance requesting a new sentence completion prediction (**setPrefix**).
- The CAT server uses the stored search graph to compute a new prediction and passed it back to the GUI (**setPrefixResult**).
- The GUI displays the new prediction to the user.
- Eventually, the user leaves the segment.
- The GUI sends a **endSession** request to the CAT tool.
- The CAT server discards all temporary data structures.
- The GUI and CAT server disconnect the web socket connection.

5.2. Machine Translation Server

For many of the CAT server's functions, information from the Machine Translation (MT) server is required. This includes not only the translation of the input sentence, but also n-best lists, search graphs, word alignments, etc.

The main call to the server is a request for a translation. The request includes the source sentence, source and target language, and optionally a key identifying the user. The server responds to requests with a JSON object, for instance:

```
{"data":
   {"translations":
     [{"sourceText": "test",
       "translatedText": "testo",
       "tokenization": {"src": [[0, 3]],
                        "tgt": [[0, 4]]}
     }]
   }
}
```

Note that this is based on the API of Google Translate. Our server implementation extends this API in various ways, such as the provision of aforementioned additional information, requests for tokenization and detokenization, etc.

6. Installation

Instructions are available online on the CASMACAT web site[3] how to install the work-bench on a consumer-grade computer running Linux.

The CASMACAT workbench uses a standard set of tools: the Apache web server, the programming language PHP, and the MySQL database. All these tools are part of a standard Linux distribution but may need to be installed on demand. The most computationally demanding process will be training a machine translation system on large amounts of data.

6.1. Web Server

The main server component is the CASMACAT web server that runs under Apache and provides the user interface over any internet browser.

Download the Source Code   First find a suitable directory for the CASMACAT code. If you install it as a user, you can place it in your home directory. If you install it as administrator, you may choose something like `/opt/casmacat`

In that directory, type:

```
git clone git://git.assembla.com/matecat_source.git web-server
cd web-server
git checkout casmacat
```

You will find additional installation instructions in the file `INSTALL.txt`. It may be more up-to-date and contain more information than the instructions below.

Create a Database   The files `lib/model/matecat.sql` and `lib/model/casmacat.sql` contain the configuration for the database. You may want to edit these files to change the name of the database, which by default is `matecat_sandbox`. If you do so, please change both files. You will also need to set up a user for the database. There may be a GUI in your Linux distribution to make this easy, otherwise you can call MySQL from the command line:

```
mysql -u root -p
mysql> connect mysql;
mysql> create user johndoe@localhost identified by 'secretpw';
mysql> create database matecat_sandbox;
mysql> grant usage on *.* to johndoe@localhost;
mysql> grant all privileges on matecat_sandbox.* to johndoe@localhost;
```

---

[3]`http://www.casmacat.eu/index.php?n=Workbench.Workbench`

With user account in place and (possibly edited) configuration files, you can now set up the database:

```
mysql -u johndoe -p < lib/model/matecat.sql
mysql -u johndoe -p < lib/model/casmacat.sql
```

To complete the setup of the database, you have to create a copy of the web server configuration file and edit it to point to your database installation.

Set Up the Web Server   First, you need to create a user account (such as `catuser`) and add it to the `www-data` group (as root). Apache needs to be configured to access your CASMACAT web server installation. This is done with a configuration file in `/etc/apache2/sites-available`, linked to from `/etc/apache2/sites-enabled`. This configuration file follows a template provided in the source directory. With all this, you may now restart Apache with `apache2ctl restart`. If you now point your web browser to your site, you should see the CASMACAT home page.

Test the Installation   To use the tool, you will have to set up a CAT server. We describe in the next section how to do this. If you want to test your current setup, you can also use a demo CAT server at the University of Edinburgh. The installation web page shows provides the configuration files and a test document that can be used for translation.

6.2. CAT Server

The computer aided translation (CAT) server communicates with the machine translation server to provide services to the CASMACAT Workbench.

Install   The CAT server is available at the following Git repository:

```
cd /opt/casmacat
git clone git://github.com/hsamand/casmacat-cat-server.git cat-server
```

Configure   Currently, the CAT server is set up to only serve one language pair (and system). It calls the MT server with a HTTP request.

The configuration of the URL of the machine translation server is currently hardcoded in lines 103–106 of `cat-server.py`:

```
port = 8644
if isinstance (text, unicode):
  text = text.encode ('UTF-8')
url = 'http://127.0.0.1:%d/%s?%s' % (
```

Please change these lines if you machine translation server does not reside on the same machine (127.0.0.1) or responds to a different port (8644).

Run   After setting the port of the machine translation server, you can run the CAT server by specifying the port it itself is listening to `./cat-server.py 9997`.

6.3. MT Server

The CASMACAT workbench may interact with any machine translation system that responds to the API according to the specifications. In the following, we describe how to set up a machine translation server using the open source Moses system. The installation requires two parts: (1) the core Moses system, from which we will use the `mosesserver` process, and (2) a Python Server Wrapper that calls `mosesserver`, alongside other pre-processing and post-processing steps, and provides additional services.

Install the Moses Server Wrapper   You can download the server script (written in Python) from its Git repository:

```
cd /opt/casmacat
git clone git://github.com/christianbuck/matecat_util.git mt-server
```

The server (in `python_server/server.py`) requires `cherrypy` to run, so you may have to install that as well.

Install Moses   Installing Moses may be a longer process, please refer to the Moses web site for installation instructions. You will need `bin/mosesserver` and various scripts in the `scripts` directory to handle pre- and post-processing.

Set Up a Toy Model   You can download a toy French-English system from the CASMACAT web site. The system consists of a model (`toy-fr-en`) directory and a number of support scripts (in the `scripts` directory). The system comes with the shell script `TOY.fr-en` that starts up `mosesserver` and the Python Moses server wrapper.

This script starts `mosesserver` to listen to port 9998 and the Python server wrapper to listen to port 9999. While `mosesserver` carries out the core translation task, the Python server wrapper deals with additional pre- and post-processing.

Connect the MT Server to the CASMACAT Workbench   To point your CAT server to your machine translation server, you have to edit in `cat-server.py` the following lines:

```
port = 9999
if isinstance (text, unicode):
  text = text.encode ('UTF-8')
url = 'http://127.0.0.1:%d/%s?%s' % (
```

Now restart your CAT server with `./cat-server.py 9997`. You have completed the installation of the CASMACAT Workbench.

7. Outlook

This public release of the workbench occurs at the mid-point of the project and offers basic functionality of the main components of the system. For the remainder of the project, a number of extensions will be integrated, such as the visualization of translation options, a bilingual concordancer, paraphrasing on demand. We also expect that several of the capabilities will be refined and their quality improved.

In collaboration with the MATECAT project we also expect the implementation of functionality targeted at professional users, such as better user administration and document management.

Acknowledgements

Bibliography

Alabau, Vicent, Luis A. Leiva, Daniel Ortiz-Martínez, and Francisco Casacuberta. User evaluation of interactive machine translation systems. In *Proc. EAMT*, pages 20–23, 2012.

Barrachina, Sergio, Oliver Bender, Francisco Casacuberta, Jorge Civera, Elsa Cubel, Shahram Khadivi, Antonio Lagarda, Hermann Ney, Jesús Tomás, Enrique Vidal, and Juan-Miguel Vilar. Statistical approaches to computer-assisted translation. *Computational Linguistics*, 35 (1):3–28, 2009.

Brown, Peter F, Vincent J Della Pietra, Stephen A Della Pietra, and Robert L Mercer. The mathematics of statistical machine translation: Parameter estimation. *Computational linguistics*, 19(2):263–311, 1993.

Federico, Marcello, Alessandro Cattelan, and Marco Trombetti. Measuring user productivity in machine translation enhanced computer assisted translation. In *Proceedings of the Tenth Conference of the Association for Machine Translation in the Americas (AMTA)*, 2012. URL `http://www.mt-archive.info/AMTA-2012-Federico.pdf`.

González-Rubio, Jesús, Daniel Ortiz-Martínez, and Francisco Casacuberta. On the use of confidence measures within an interactive-predictive machine translation system. In *Proc. EAMT*, 2010.

Koehn, Philipp. Enabling monolingual translators: post-editing vs. options. In *Proc. NAACL*, pages 537–545, 2010.

Langlais, Philippe, George Foster, and Guy Lapalme. TransType: a computer-aided translation typing system. In *NAACL Workshop: EmbedMT*, pages 46–51, 2000.

Leiva, Luis A., Vicent Alabau, and Enrique Vidal. Error-proof, high-performance, and context-aware gestures for interactive text edition. In *Proceedings of the 2013 annual conference extended abstracts on Human factors in computing systems (CHI EA)*, pages 1227–1232, 2013.

Plitt, Mirko and Francois Masselot. A productivity test of statistical machine translation post-editing in a typical localisation context. *Prague Bulletin of Mathematical Linguistics*, 93:7–16, 2010. URL http://ufal.mff.cuni.cz/pbml/93/art-plitt-masselot.pdf.

Pouliquen, Bruno, Christophe Mazenc, and Aldo Iorio. Tapta: A user-driven translation system for patent documents based on domain-aware statistical machine translation. In Forcada, Mikel L., Heidi Depraetere, and Vincent Vandeghinste, editors, *Proceedings of th 15th International Conference of the European Association for Machine Translation (EAMT)*, pages 5–12, 2011.

Skadiņš, Raivis, Maris Puriņš, Inguna Skadiņa, and Andrejs Vasiļjevs. Evaluation of SMT in localization to under-resourced inflected language. In Forcada, Mikel L., Heidi Depraetere, and Vincent Vandeghinste, editors, *Proceedings of the 15th International Conference of the European Association for Machine Translation (EAMT)*, pages 35–40, 2011.

Address for correspondence:
Philipp Koehn
pkoehn@inf.ed.ac.uk
Informatics Forum 4.19
10 Crichton Street, Edinburgh
EH8 9AB, United Kingdom