Advanced Computer Aided Translation with a Web-Based Workbench

Vicent Alabau[⋆], Ragnar Bonk[†], Christian Buck[‡], Michael Carl[†], Francisco Casacuberta[⋆]
Mercedes García-Martínez[†], Jesús González[⋆], Philipp Koehn[‡], Luis Leiva[⋆]
Bartolomé Mesa-Lao[†], Daniel Ortiz[⋆], Herve Saint-Amand[‡], Germán Sanchis[⋆], and Chara Tsoukala[‡]
[⋆]Institut Tecnològic d'Informàtica, Universitat Politècnica de València, Spain
[†]Copenhagen Business School, Department of International Business Communication, Denmark
[‡]School of Informatics, University of Edinburgh, Scotland
valabau@iti.upv.es, ragnar.bonk@gmx.de, cbuck@lantis.de, mc.isv@cbs.dk, fcn@iti.upv.es
mgarcia@iti.upv.es, jegonzalez@dsic.upv.es, pkoehn@inf.ed.ac.uk, luileito@iti.upv.es
bm.ibc@cbs.dk, dortiz@iti.upv.es, hsamand@inf.ed.ac.uk, gsanchis@dsic.upv.es, x.tsoukala@gmail.com

Abstract

We describe a web-based 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. It is available open source and integrates with multiple MT systems.

The goal of the CASMACAT project[1] is to develop an advanced computer aided translation workbench. At the mid-point of the 3-year project, we release this tool as open source software. It already includes a wide range of novel advanced types of assistance and other functionalities that do not exist together in any other computer aided translation tool.

The CASMACAT is working in close collaboration with the MATECAT project[2], which also has the goal of developing a new open source web-based computer aided translation tool, and focuses mainly on post-editing machine translation, adaptation methods, and ease of use that make such a tool suitable for professional users.

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.

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).

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

1 Related Work

There is increasing evidence for productivity gains of professional translators when they post-edit machine translation output.

For instance, Plitt and Masselot (2010) compare post-editing machine translation against unassisted translation in a web-based tool for a number of language pairs, showing productivity gains of up to 80%. Skadiņš et al. (2011) show a 30 percent increase for English-Latvian translation with a slight but acceptable degradation in quality. Federico et al. (2012) assess the benefit of offering machine translation output in addition to translation memory matches (marked as such) in a realistic work environment for translators working on legal and information technology documents. They observe productivity gains of 20-50%, roughly independent from the original translator speed and segment length, but with different results for different language pairs and domains. Moreover, Pouliquen et al. (2011) show that, aided by machine translation, non-professional post-editors may be able to create high-quality translations, comparable to a professional translation agency.

So far, usage of machine translation technology has concentrated on human-computer interaction involving the human translator as a post-editor, but rarely involves the human translator influencing the decisions of the machine translation system. Recent efforts on building interactive machine translation systems include work by Langlais et al. (2000) and Barrachina et al. (2009). Both studies develop research systems looking into a tighter integration of human translators in MT processes by developing a prediction model that interactively suggests translations to the human translator as he or she types. Related work displays several word and phrase translation choices to human translators (Koehn, 2010).

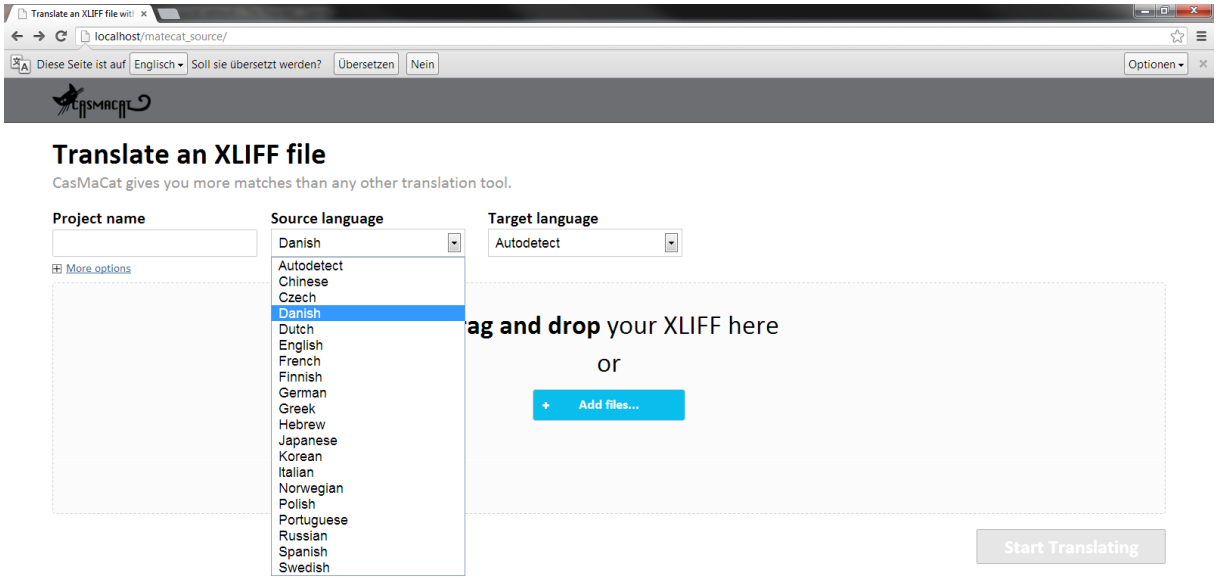

Figure 1: View for uploading new documents

2 User Interface

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.

2.1 Upload

If the user opens the default URL without giving any special parameters, he or she is taken to the upload view. This is currently the entry point of the application. See Figure 1 for a screenshot. 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 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 he or she is taken to the translate view and can start working.

2.2 Post-Editing

In the translate view, the user can now translate the document (see Figure 2). 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 status to a segment, for instance, *translated* for finished ones or *draft* for segments, where he or she is not yet sure about the translation and he or she wants to review later. When finished, the *Download Project* button may be used to download the translated document, again in the XLIFF format.

When in the translate view, all the actions of the user that are related to the translation task, e.g. typing, choosing a suggestion, closing a segment and so on, are logged by the CASMACAT logging module. In addition to traditional key and mouse logging, we also provide text change logging based on the HTML5 *input* element. This makes the log of text activities much more robust, e.g. it allows to log changes from paste or cut actions triggered by the browser's menu bar or the context menu of the mouse. Mouse clicks are still logged to track user interactions with UI elements. Key logging is helpful for offline analysis.

2.3 Interactive Translation Prediction

In the following paragraphs we present a short description of the main advanced CAT features that we implemented in the workbench. Such features

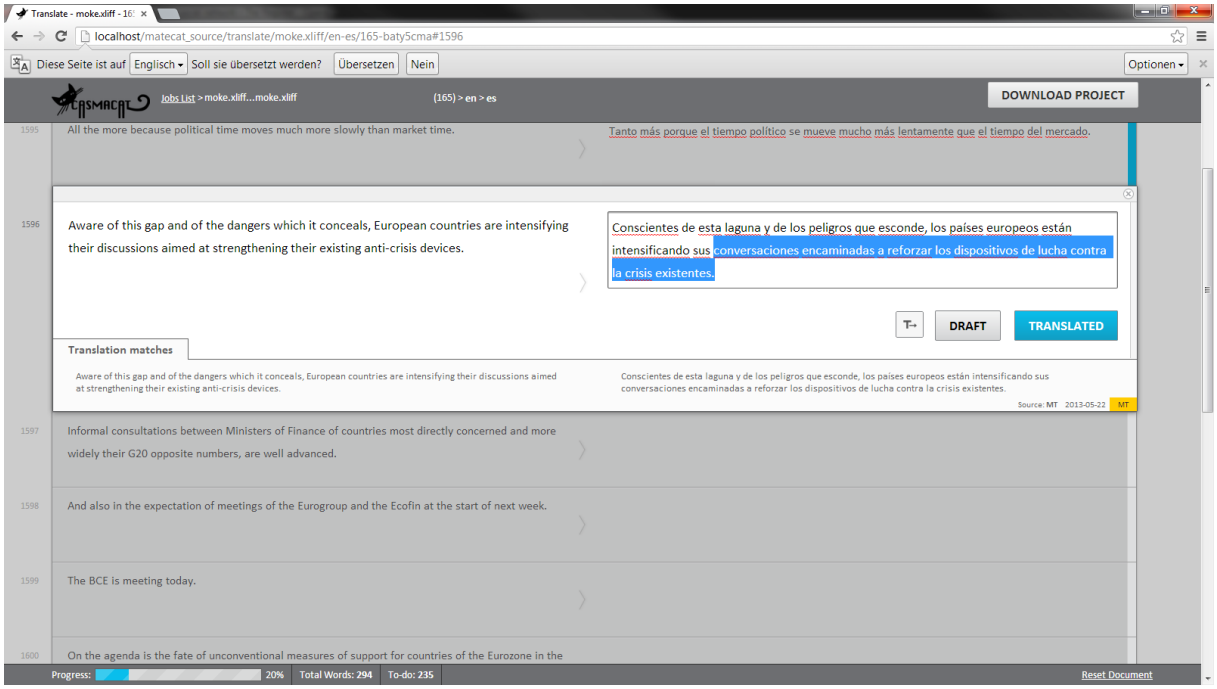

Figure 2: Translate view with post-editing configuration

are different in nature, but all of them aim at boosting translator productivity.

Intelligent Autocompletion Interactive translation prediction takes place 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.

Confidence Measures Confidence measures (CMs) have two main applications in MT (González-Rubio et al., 2010). Firstly, CMs allow the user to spot wrong translations (for instance, by painting in red those translations with very low confidence). Secondly, CMs can also inform the user about the translated words that are possibly incorrect, but still have a chance of being correct (for instance, painted in orange). In our workbench, both applications are handled by means of two thresholds, one that favors precision and another that favors recall of changes to highlighted words.

We use confidence measures to inform the user about translation reliability 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

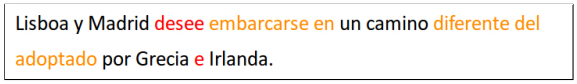

Figure 3: Visualization of Confidence Measures

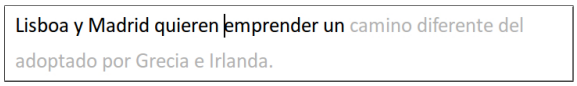

Figure 4: Interactive Translation Prediction

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. See Figure 3 for a screenshot of the text highlighting in the edit area.

Prediction Length 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 the CMs.

In our implementation, pressing the Tab key allows the user to ask the system for the next set of predicted words. See Figure 4 for a screenshot.

Search and Replace Most of the computer-assisted translation tools provide the user with intelligent search and replace functions for fast text

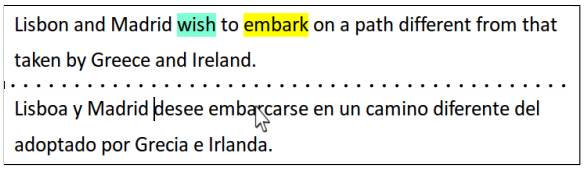

Figure 5: Visualization of Word Alignment

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.

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. See Figure 5 for a screenshot.

Prediction Rejection With the purpose of easing user interaction, our workbench also supports a one-click rejection feature (Sanchis-Trilles et al., 2008). This invalidates the current prediction for the sentence that is being translated, and provides the user with an alternate one, in which the first new word is different from the previous one.

2.4 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. This capability is explained in detail in Section 4. 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 he or she interacted with the tool.

2.5 List Documents

Another view details a list of documents submitted to the tool. From there a user can start a replay, download the logged data or continue a translation session.

3 Server

The overall design of the CASMACAT workbench is very modular. There are three independent components (see also Figure 6): the GUI/web server, the CAT server and the MT server.

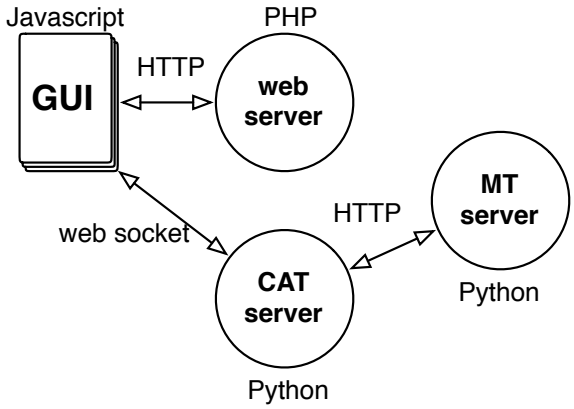

Figure 6: Modular design of the workbench: Web-based components (GUI and web server), CAT server and MT server are independent and can be swapped out

We separate these components by clearly specified API calls, so that alternative implementations can be used as well. We expect that the CASMACAT workbench may be use partially, for instance in the following fashion:

- As part of a larger localization workflow with existing editing facilities, only the capabilities of the CASMACAT CAT server and CASMACAT MT server are used. A legacy editing tool is extended to make calls to the CAT server and thus benefit from additional functionality.
- If an existing customized MT translation solution is already in place, then the CASMACAT front-end and CAT server can connect to it.

Already, the currently implemented CASMACAT workbench supports two different MT server components, Moses (Koehn et al., 2007) and Thot (Ortiz-Martínez et al., 2005).

3.1 CAT Server

The 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.

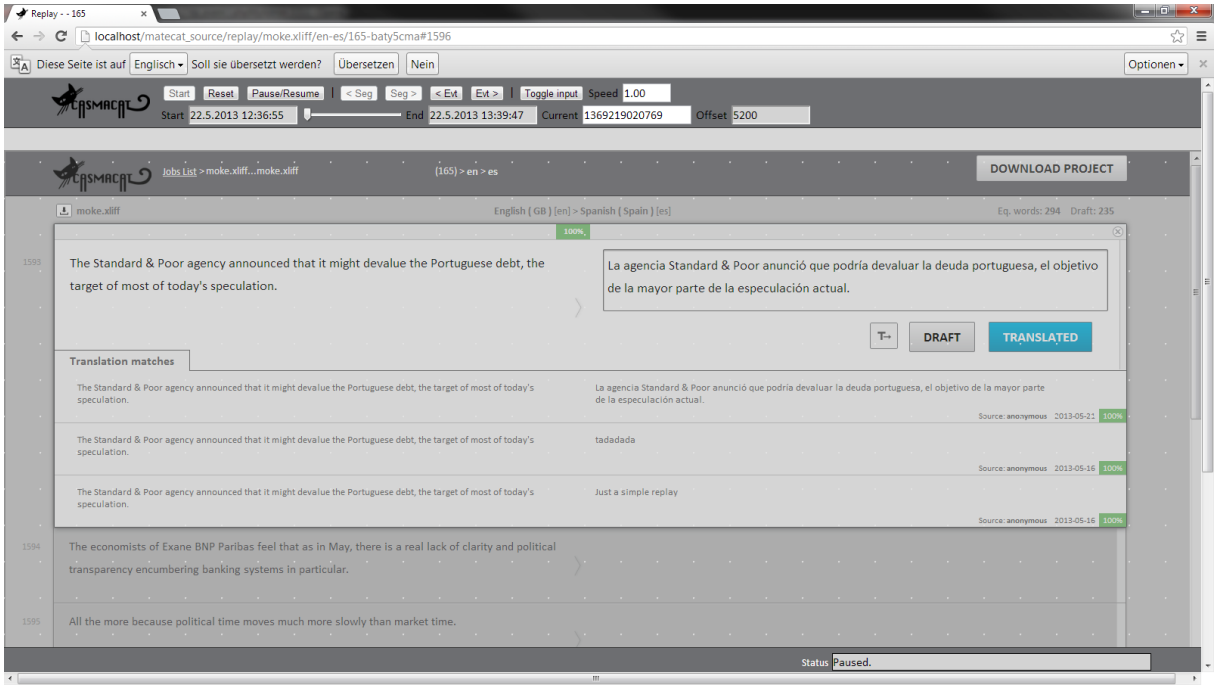

Figure 7: Replay view

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.

The interaction between the GUI and the CAT server follows a well-defined API.

3.2 MT 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 an 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.

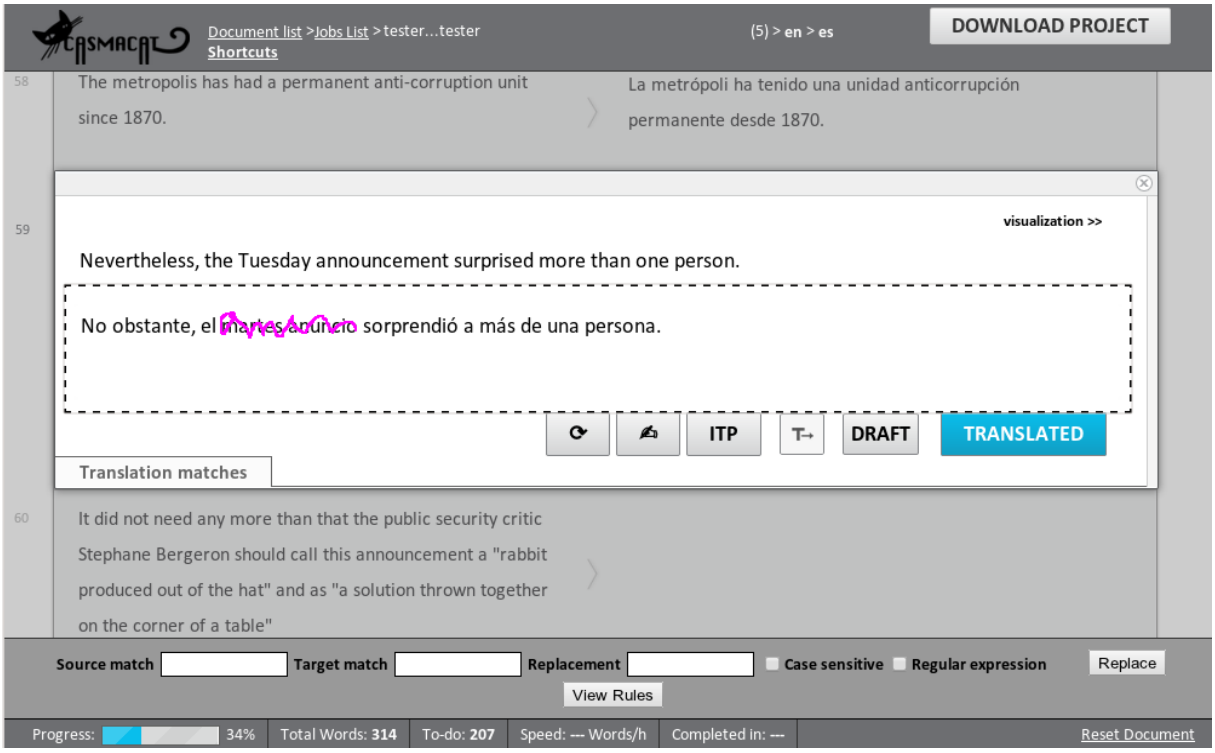

Figure 8: Sketch of a document fragment

4 Replay Mode for User Activity Data

The replay view (see Figure 7) loads the translate view into an *iframe* and remote-controls it with the data from the log file. The session appears in the web browser exactly the same way as it appeared to the user interacting with the tool. The current implementation is robust to user actions, for instance it allows for changes of the replaying window geometry (like resizing).

The log data is fetched in small chunks when replaying. Upon start-up, only the first chunk of the log data is loaded and the replay starts. When the next chunk is needed, the replay is paused and the next chunk is fetched. This minimizes the initial loading time when starting the replay.

The replay engine uses precise internal clocking. Each event is replayed on its on. This makes the engine precise and robust and allows for arbitrary jumps between events.

Many events are included in the logging (such as all events around interactive translation prediction). The functionality is currently being extended to allow for arbitrary seeking in the replay (e.g. by time or segment). Additionally, the replay mode will soon allow to re-compute or re-map particular data, like gaze-to-char mapping.

Latest tests have confirmed that the current strategy of visualizing the eye tracking data via the browser's DOM is too slow. The new idea is to let the eye tracking plugin take over this task by creating a new native but invisible system window on which the eye tracking data is drawn. This still has to be implemented and tested but promises a high performance visualization of eye tracking data.

5 E-pen Interaction

E-pen interaction should be regarded as a complementary input rather than a complete replacement of the keyboard. In a first approach, we have extended the CASMACAT UI with the minimum components necessary to enable e-pen gestures and handwriting in a comfortable way.

5.1 E-pen UI

When the e-pen UI is enabled, a new button is displayed in the button area (✍). This button toggles the e-pen view. When activated, the display of the current segment is changed so that the source segment is shown above the target segment. This way, the drawing area is maximized horizontally, which facilitates handwriting particularly in tablet

devices.

Next, an HTML canvas element is added over the target segment. This drawing area is highlighted with a dashed border. In addition, a clear button () is added to refresh the drawing area. A screenshot of such display can be seen in Figure 8.

The user can interact with the system by writing on the canvas. 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.

Thus, we decided to use MINGESTURES (Leiva et al., 2013), a highly accurate, high-performance gestures for interactive text editing. The gestures in MINGESTURES are defined by 8 straight lines that can be configured to be direction dependent and be aware of the context where they gestures takes place. In addition, they can be easily differentiated from handwritten text with line fitting algorithms. Gestures are recognized in the client side so the response is almost immediate.

Conversely, when handwritten text is detected, the pen strokes are sent to the server. At this moment, only single words can be written. However, in future releases also substrings and multiple words will be allowed. The set of gestures used in the workbench are summarized in Figure 9.

5.2 HTR server

The hand-written text recognition (HTR) server is responsible for decoding the user handwriting into digital text. The technology is based very much on the ITP server technology. An HTR server must implement the following API:

startSession This function instructs the server to initialize a new HTR session with the appropriate contextual information. A session consists of one or more strokes that constitute one user interaction. The input parameters are the **source** string, the current **translation** and the last position validated by the user. At this stage, the server does not return a value.

addStroke When a user finishes writing a stroke, the points are encoded into an array of points that are defined by the x and y coordinates along with the *timestamp* when they were acquired. The HTR server processes this information and, optionally, returns a partial decoding.

endSession When the user stops writing for a specific amount of time (400ms in our set-up), the users session finishes. The final decoding is then returned to the UI, possibly with a list of n-best solutions.

The HTR server is based on iAtros, an open source HMM decoder. The current version does not leverage contextual information, but it is prepared to support that in future releases.

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

Analyzing the eye tracking data requires a tool for aligning and correcting erroneous fixations, to manually map them on the words that are likely to be fixated. The tool is instrumental in creating a corpus of high-quality gaze-to-word mappings, and is used as a training set of automatic gaze-to-word mappings algorithms. A first version of the manual alignment tool was implemented and we also implemented two algorithms for automatic gaze-to-word alignment.

One way to visualize the eye tracking data is by plotting translation sessions in the form of translation progression graphs. However, translation progression graphs only visualize a small fraction of the information.

7 Outlook

We are currently conducting a field trial to extensively test the workbench in a real-world environ-

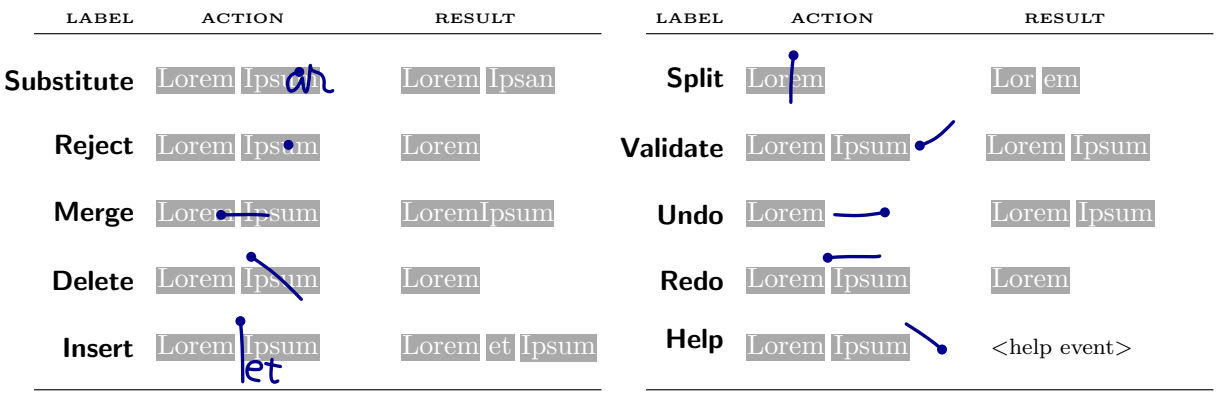

Figure 9: Set of gestures

ment of professional translators. We will collect extensive logging information that will allow analysis of translator behavior and inform the future development of the various technologies.

We hope that by releasing the CASMACAT workbench open source, the broader research community can carry out similar studies.

References

Alabau, V., Leiva, L. A., Ortiz-Martínez, D., and Casacuberta, F. (2012). User evaluation of interactive machine translation systems. In *Proc. EAMT*, pages 20–23.

Barrachina, S., Bender, O., Casacuberta, F., Civera, J., Cubel, E., Khadivi, S., Lagarda, A., Ney, H., Tomás, J., Vidal, E., and Vilar, J.-M. (2009). Statistical approaches to computer-assisted translation. *Computational Linguistics*, 35(1):3–28.

Brown, P. F., Pietra, V. J. D., Pietra, S. A. D., and Mercer, R. L. (1993). The mathematics of statistical machine translation: Parameter estimation. *Computational linguistics*, 19(2):263–311.

Federico, M., Cattelan, A., and Trombetti, M. (2012). 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)*.

González-Rubio, J., Ortiz-Martínez, D., and Casacuberta, F. (2010). On the use of confidence measures within an interactive-predictive machine translation system. In *Proc. EAMT*.

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

Koehn, P., Hoang, H., Birch, A., Callison-Burch, C., Federico, M., Bertoldi, N., Cowan, B., Shen, W., Moran, C., Zens, R., Dyer, C. J., Bojar, O., Constantin, A., and Herbst, E. (2007). Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions*, pages 177–180, Prague, Czech Republic. Association for Computational Linguistics.

Langlais, P., Foster, G., and Lapalme, G. (2000). TransType: a computer-aided translation typing system. In *NAACL Workshop: EmbedMT*, pages 46–51.

Leiva, L. A., Alabau, V., and Vidal, E. (2013). 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.

Ortiz-Martínez, D., García-Varea, I., and Casacuberta, F. (2005). Thot: a toolkit to train phrase-based statistical translation models. In *Proceedings of the Tenth Machine Translation Summit (MT Summit X)*, Phuket, Thailand.

Plitt, M. and Masselot, F. (2010). A productivity test of statistical machine translation post-editing in a typical localisation context. *Prague Bulletin of Mathematical Linguistics*, 93:7–16.

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

Sanchis-Trilles, G., Ortiz-Martínez, D., Civera, J., Casacuberta, F., Vidal, E., and Hoang, H. (2008). Improving interactive machine translation via mouse actions. In *Proc. EMNLP*.

Skadiņš, R., Puriņš, M., Skadiņa, I., and Vasiļjevs, A. (2011). Evaluation of SMT in localization to under-resourced inflected language. In Forcada, M. L., Depraetere, H., and Vandeghinste, V., editors, *Proceedings of the 15th International Conference of the European Association for Machine Translation (EAMT)*, pages 35–40.