D5.3: Beta release of CASMACAT workbench

Ragnar Bonk, Vicent Alabau, Michael Carl, Philipp Koehn

Distribution: Public

CASMACAT
Cognitive Analysis and Statistical Methods
for Advanced Computer Aided Translation

ICT Project 287576 Deliverable D5.3

Project funded by the European Community
under the Seventh Framework Programme for
Research and Technological Development.

Project ref no.	ICT-287576
Project acronym	CASMACAT
Project full title	Cognitive Analysis and Statistical Methods for Advanced Computer Aided Translation
Instrument	STREP
Thematic Priority	ICT-2011.4.2 Language Technologies
Start date / duration	01 November 2011 / 36 Months

Distribution	Public
Contractual date of delivery	April 30, 2013
Actual date of delivery	June 13, 2013
Date of last update	June 13, 2013
Deliverable number	D5.3
Deliverable title	Beta release of CASMACAT workbench
Type	Report
Status & version	Draft
Number of pages	42
Contributing WP(s)	WP5
WP / Task responsible	UEDIN, CBS, UPV
Other contributors	
Internal reviewer	Philipp Koehn
Author(s)	Ragnar Bonk, Vicent Alabau, Michael Carl, Philipp Koehn
EC project officer	Aleksandra Wesolowska
Keywords	

Executive Summary

This document contains details about the implementation of the 2nd prototype of the CASMACAT workbench and the Translation Process Research Database (TPR-DB). It outlines the major components of the workbench and their usage (Sections 1, 2, 3 and 6), as well as the structure and feature of the TPR-DB (Section 7). Since gaze information is the most valuable source for tracking translator effort in text understanding, and due to the noise inherent in current head-free eye-tracking technology, Sections 4 and 5 report attempts to implement solutions for obtaining better gaze-to-word mapping accuracy.

At the time of this writing, an installation guide[1] has been written and made available to a select group of alpha testers (researchers from universities and research laboratories) to prepare a wider release of the prototype.

Contents

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

1 T5.2 Graphical Interface

This section describes the current state of the GUI of the 2nd CASMACAT prototype. It deals with the new GUI design and functionality taken over from MATECAT and the CASMACAT extensions that have been added.

To fulfill the requirement of a collaboration between the CASMACAT and the MATECAT projects, attempts have been made to merge both projects into one common code base. After analysing the capabilities of the MATECAT UI and testing its compatibility with CASMACAT requirements, especially eye tracking (ET), the decision has been made to adapt the MATECAT code and extend it with the CASMACAT features. The goal is to get a common UI where particular functionality can be switched on or off as needed. Additionally, test results of field trials which will be carried out in the two projects will be better comparable with a common UI.

Even though there clear advantages in a MATECAT and CASMACAT collaboration, there is also a price to pay: CASMACAT's 1st prototype already had important functionality (especially multi-user capabilities), that had to be dropped and is not yet implemented in MATECAT. Also, requirements and handling of data is in some cases very different, as for instance the format of a translation in the database: MATECAT stores additional formatting information (HTML and XLIFF) and uses the encoding given in the original document while CASMACAT stores UTF-8 encoded pure text. If and how this can be normalized, is still a challenge to face.

1.1 The new UI

The new CASMACAT / MATECAT UI consists of a couple of views designated to different tasks. Of course, the translate view is the 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 (re-)implemented by re-using the knowledge gathered in the 1st prototype. The different views will now be shown and described in the sequence they are typically used.

1.1.1 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 the XLIFF format, the only currently supported format. The language can either be chosen manually or auto-detected from the XLIFF file. Not all entries in the drop-down menu have a working MT engine behind them. If several documents are uploaded at once, they are bundled into one job and are translated in a sequence. Note that for CASMACAT it is per definition not allowed to upload several documents. This would make logging and replay more complex and would (especially with ET) create such a huge amount of data, that replay would become impossible. If the user clicks on the 'Start Translating' button he or she is taken to the translate view and can start working.

1.1.2 Translate (PE)

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,

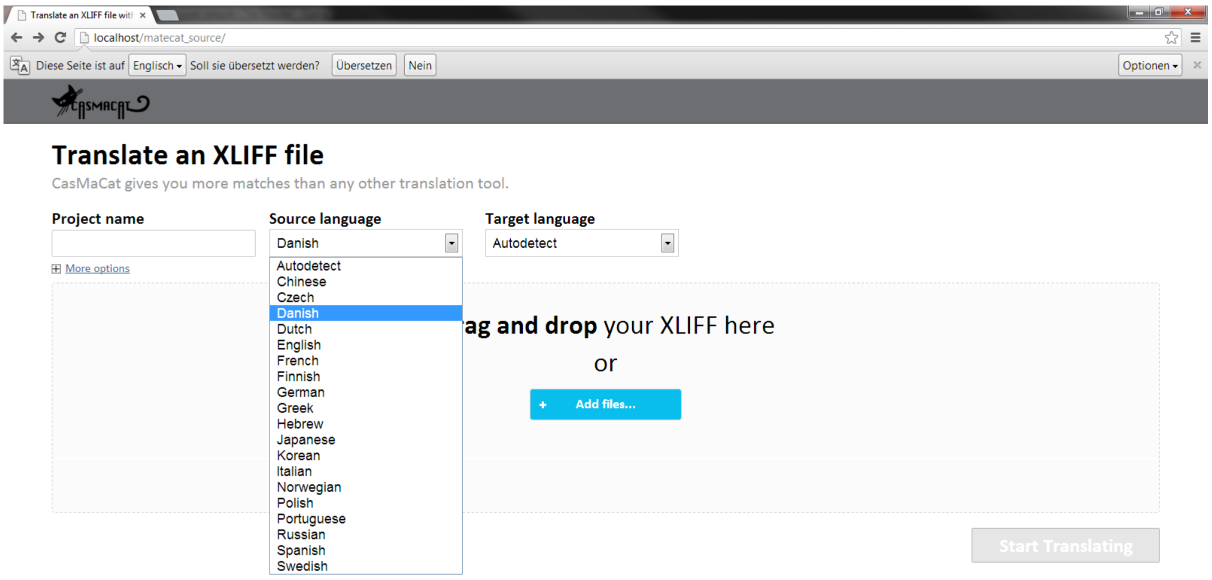


Figure 1: View for uploading new documents

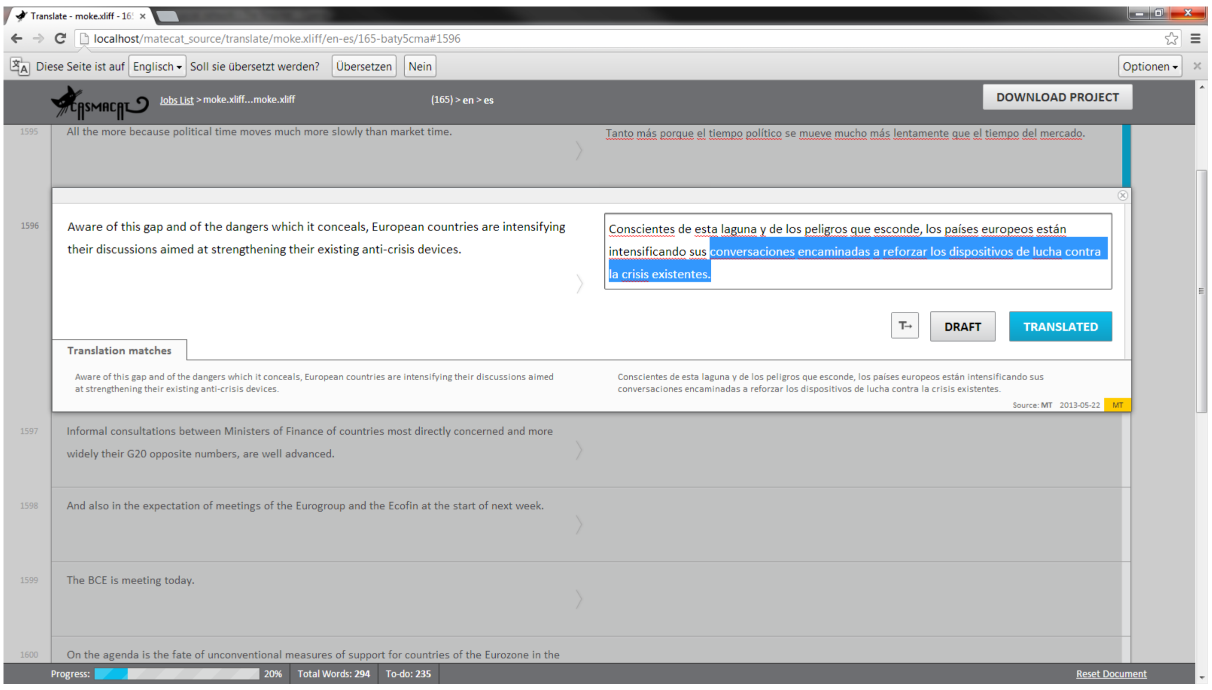
for instance, to go to the next segment or to copy the source text to the target. The user can assign different states to a segment, for instance, 'translated' for finished ones or 'draft' for segments, where he or she is not yet sure about the translation and where he or she wants to review it again later. When finished, the 'Download Project' button may be used to download the translated document, again in the XLIFF format.

When in this 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. This logging module has been completely re-written from scratch, based on the knowledge of the 1st prototype and implementing new better techniques for more reliable log data. Traditional key and mouse logging has been fully replaced by 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. Of course, mouse clicks are still logged to track user interactions with UI elements. Key logging is also still available and running in parallel by default. It is optional now and not required (for replay), but for offline analysis, it may still be of interest (e.g. calculate typing rate).

1.1.3 Translate (ITP)

In the following paragraphs we present a short description of the main features that were implemented in the prototype, which are summarized in Figure 3. Such features are different in nature, but all of them aimed at boosting translator productivity.

Intelligent Autocompletion IMT takes place every time a keystroke is detected by the system [2]. In such event, the system produces a (full) suitable prediction according to the text that the user is writing. This new prediction replaces the remaining words of the original sentence at the right of the text cursor.


Translate - moke.xliff - 16
localhost/matecat_source/translate/moke.xliff/en-es/165-baty5cma#1596
Diese Seite ist auf Englisch Soll sie übersetzt werden? Übersetzen Nein
Optionen
Jobs List > moke.xliff...moke.xliff
(165) > en > es
DOWNLOAD PROJECT
DRAFT
TRANSLATED
Translation matches
Source: MT 2013-05-22 MT
Progress: 20% Total Words: 294 To-do: 235
Reset Document

1595 All the more because political time moves much more slowly than market time.

Tanto más porque el tiempo político se mueve mucho más lentamente que el tiempo del mercado.

1596 Aware of this gap and of the dangers which it conceals, European countries are intensifying their discussions aimed at strengthening their existing anti-crisis devices.

Conscientes de esta laguna y de los peligros que esconde, los países europeos están intensificando sus conversaciones encaminadas a reforzar los dispositivos de lucha contra la crisis existentes.

Aware of this gap and of the dangers which it conceals, European countries are intensifying their discussions aimed at strengthening their existing anti-crisis devices.

Conscientes de esta laguna y de los peligros que esconde, los países europeos están intensificando sus conversaciones encaminadas a reforzar los dispositivos de lucha contra la crisis existentes.

1597 Informal consultations between Ministers of Finance of countries most directly concerned and more widely their G20 opposite numbers, are well advanced.

1598 And also in the expectation of meetings of the Eurogroup and the Ecofin at the start of next week.

1599 The BCE is meeting today.

1600 On the agenda is the fate of unconventional measures of support for countries of the Eurozone in the

Figure 2: Translate view with post-editing configuration

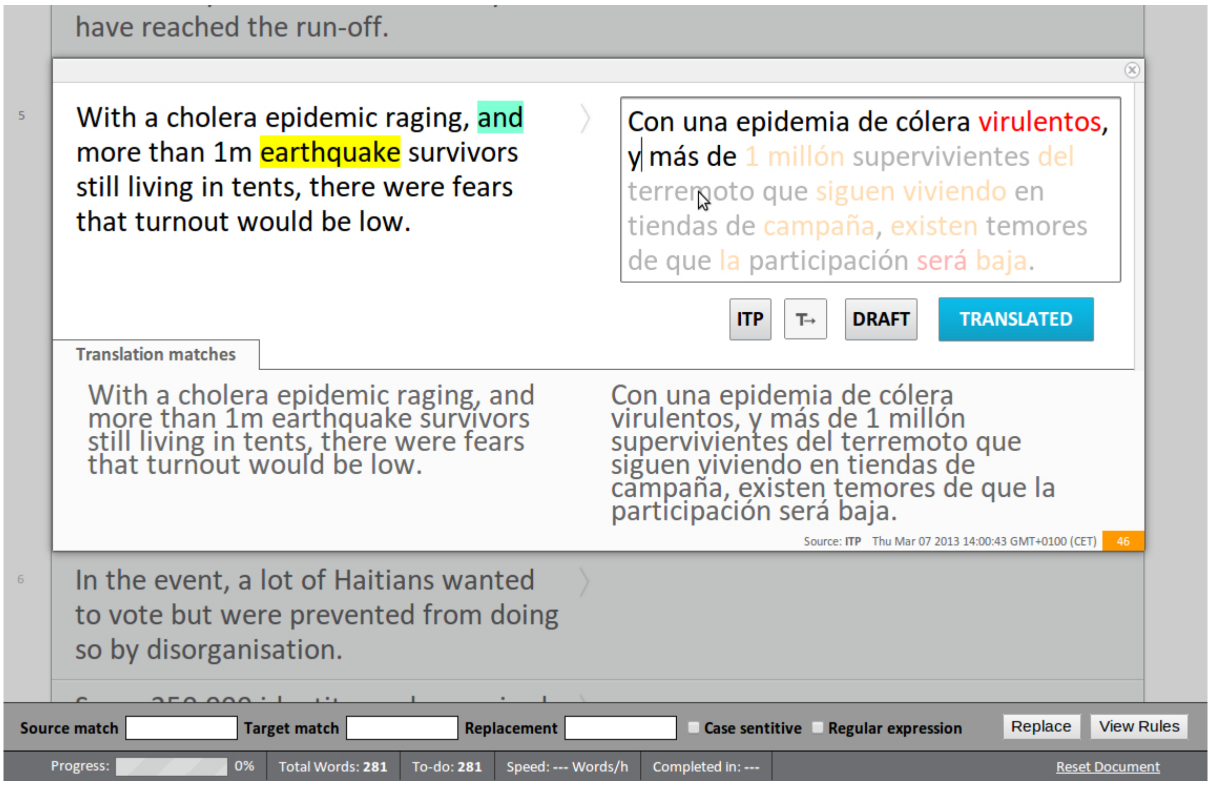
Confidence Measures In our workbench, we use confidence measures to inform the users about translation reliability under two different criteria. On the one hand, we highlight in red color those translated words that are likely to be incorrect. We use a threshold that maximizes precision in detecting 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 maximizes recall. See Figure 4 for a screenshot.

Prediction Length Providing the user with a new prediction whenever a key is pressed has been proved to be cognitively demanding [1]. Therefore, it was 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 5 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 revision. Our prototype 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. See Figure 6 for a screenshot.

Word Alignment Information Alignment of source and target information is an important part of the translation process [3]. 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 7 for a screenshot.

Prediction Rejection With the purpose of easing user interaction, our prototype also supports a one-click rejection feature [5]. 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.


have reached the run-off.

5 With a cholera epidemic raging, and more than 1m earthquake survivors still living in tents, there were fears that turnout would be low.

Con una epidemia de cólera virulentos, y más de 1 millón supervivientes del terremoto que siguen viviendo en tiendas de campaña, existen temores de que la participación será baja.

With a cholera epidemic raging, and more than 1m earthquake survivors still living in tents, there were fears that turnout would be low.

Con una epidemia de cólera virulentos, y más de 1 millón supervivientes del terremoto que siguen viviendo en tiendas de campaña, existen temores de que la participación será baja.

Source: ITP Thu Mar 07 2013 14:00:43 GMT+0100 (CET) 46

6 In the event, a lot of Haitians wanted to vote but were prevented from doing so by disorganisation.

Figure 3: Translate view with advanced ITP configuration

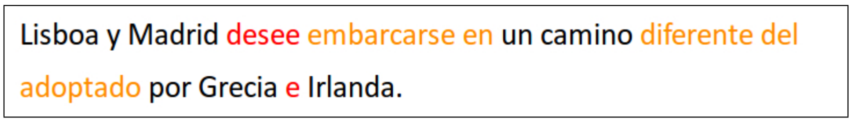

Lisboa y Madrid desee embarcarse en un camino diferente del adoptado por Grecia e Irlanda.

Figure 4: Visualization of Confidence Measures

1.1.4 Replay

The prototype 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 6. 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.

1.1.5 List Documents

As in the 1st prototype it is possible to list all documents in the system (Figure 8). From there a user can start a replay, download the logged data or continue a translation session. Unfortunately, as there is currently no real multi-user support, this view shows all documents of all users. So it is currently more an administration tool. Later, when a user management is

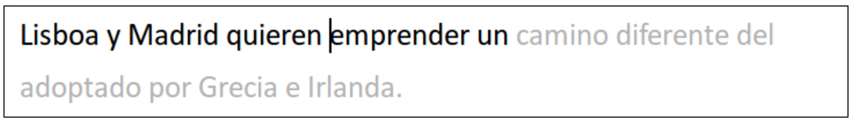

Lisboa y Madrid quieren emprender un camino diferente del adoptado por Grecia e Irlanda.

Figure 5: Interactive Translation Prediction

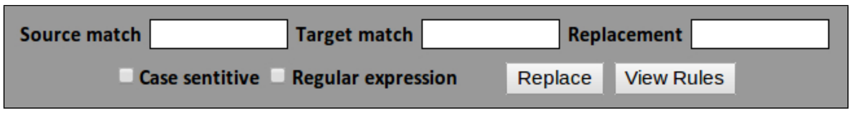
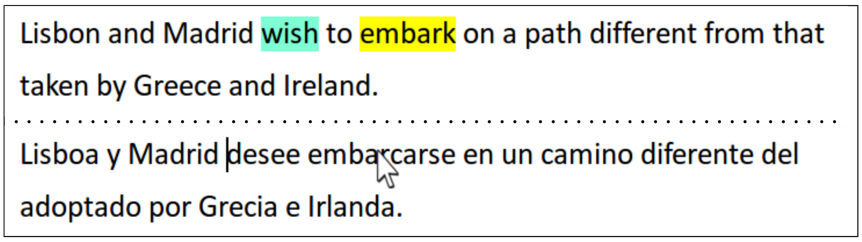


Figure 6: Interactive Translation Prediction



Lisbon and Madrid wish to embark on a path different from that taken by Greece and Ireland.

Lisboa y Madrid desee embarcarse en un camino diferente del adoptado por Grecia e Irlanda.

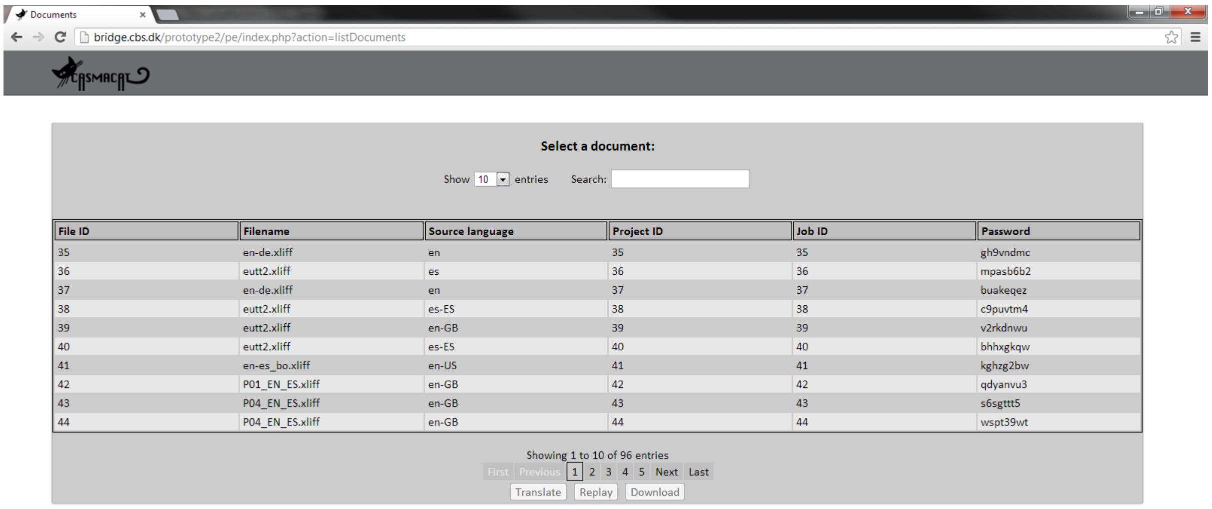
Figure 7: Visualization of Word Alignment

added, this view could present documents only 'per-user'. For this reason, the url is not exposed in one of the other views. To open this view, one needs to specify 'listDocuments' as the action:

```
http://<server>/<baseurl>/index.php?action=listDocuments
```

For the post-editing demo (see Section 1.4), this would be

```
http://bridge.cbs.dk/prototype2/pe/index.php?action=listDocuments
```



bridge.cbs.dk/prototype2/pe/index.php?action=listDocuments

Select a document:

Show 10 entries Search:

File ID	Filename	Source language	Project ID	Job ID	Password
35	en-de.xliff	en	35	35	gh9vndmc
36	eutt2.xliff	es	36	36	mpasb6b2
37	en-de.xliff	en	37	37	buakeqez
38	eutt2.xliff	es-ES	38	38	c9puvtm4
39	eutt2.xliff	en-GB	39	39	v2rkdnwu
40	eutt2.xliff	es-ES	40	40	bhhxgkqw
41	en-es_bo.xliff	en-US	41	41	kghzg2bw
42	P01_EN_ES.xliff	en-GB	42	42	qdyanvu3
43	P04_EN_ES.xliff	en-GB	43	43	s6sgttt5
44	P04_EN_ES.xliff	en-GB	44	44	wspt39wt

Showing 1 to 10 of 96 entries

Figure 8: Documents

1.2 Server

On the server side the MATECAT code has been adjusted and CASMACAT modules have been added. Integration has been carried out very carefully so as not to break existing functionality and to make it easy merging the code back into the MATECAT code base. If possible, files were put into separate directories or a prefixed was added. Function hooks have been used to extend the orignal code without altering it. If this was not possible, then changes to the original MATECAT code have been cleanly commented. Also, the additional database tables reside in an extra script and can be easily imported over the original MATECAT database.

1.3 Configuration

The system has been designed in a manner that all CASMACAT modules are optional and can be switched on or off as needed. The particular configuration can be changed in the config.ini file. Currently, the selected configuration is then applied for the whole system. All users work then with the same configuration. This implies that two instances of CASMACAT need to be run, if one would like to run two different settings. However, the structure of the current implemention opens up the possibility to easily allow a system later, where each user can have it's own setting, either configured by himself or herself or by some administrator.

1.4 Demo

The 2nd prototype can be tried out in two different modes, post-editing and ITP (with search and replace). As this is a beta version, there are no particular security checks. The application *is* vulnerable against several types of attacks (e.g. XSS) and should not be used in any production environment. The URL's to try the system are:

- `http://bridge.cbs.dk/prototype2/pe`
- `http://bridge.cbs.dk/prototype2/itp-sr`

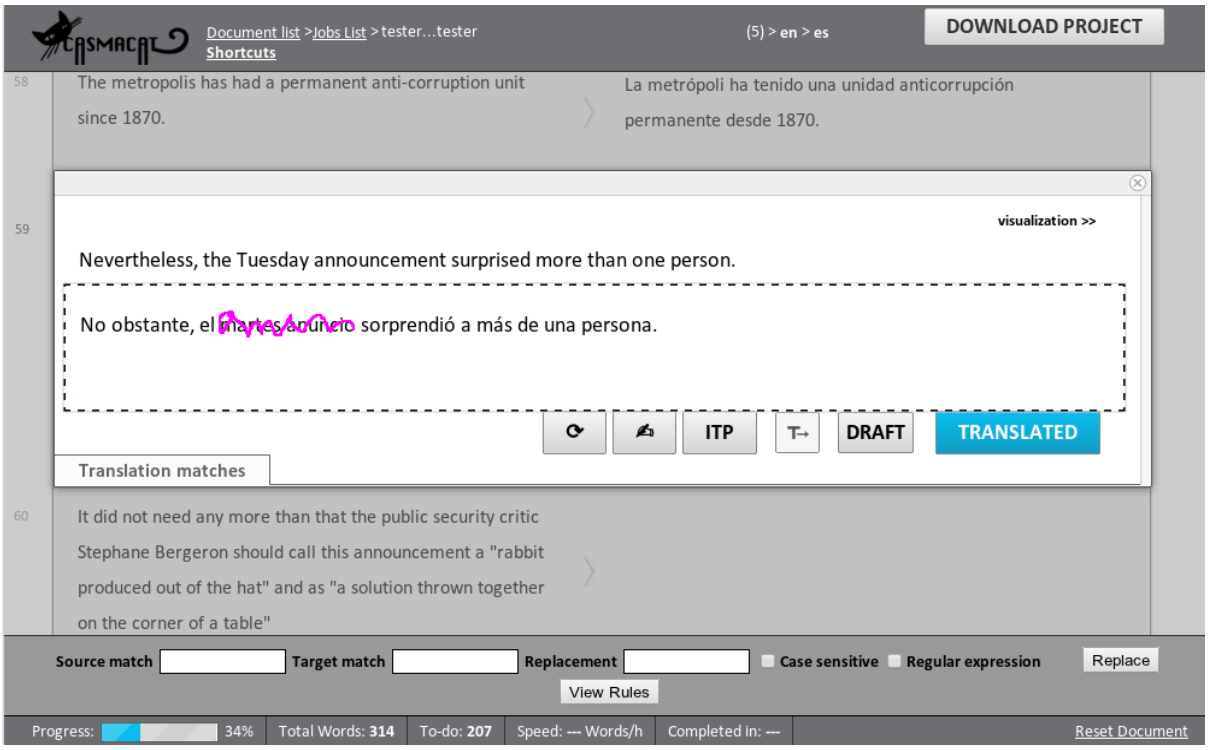
2 T5.3 E-pen Interaction

This section is devoted to the integration of the e-pen subsystem in the CASMACAT UI. E-pen interaction should be regarded as a complementary input rather than a complete replacement of the keyboard. However, this should not be an obstacle to completely redesign the original interface to better accommodate for the e-pen interaction specific needs. Nevertheless, 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.

2.1 E-pen UI

The e-pen UI can be enabled by setting `penenabled = 1` and `htrserver = "address"` in the server configuration file. As a result, 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 9.

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 decide to focus on usability. We believe that a fast response and a good



58 The metropolis has had a permanent anti-corruption unit since 1870. — La metrópoli ha tenido una unidad anticorrupción permanente desde 1870.

59 Nevertheless, the Tuesday announcement surprised more than one person.

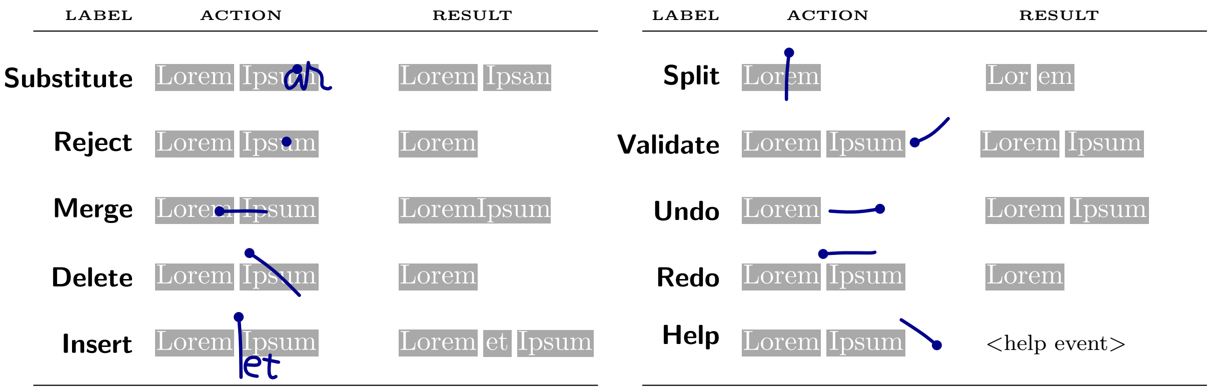
No obstante, el martes anuncio sorprendió a más de una persona.

60 It did not need any more than that the public security critic Stephane Bergeron should call this announcement a "rabbit produced out of the hat" and as "a solution thrown together on the corner of a table"

Progress: 34% | Total Words: 314 | To-do: 207 | Speed: --- Words/h | Completed in: ---

Figure 9: Sketch of a document fragment

accuracy are critical for user acceptance. Thus, we decided to use MinGestures [4], a highly accurate, high-performance, gestures for interactive text edition. 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. On the other hand, 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 prototype are summarized in Figure 10.



LABEL	ACTION	RESULT	LABEL	ACTION	RESULT
Substitute	Lorem Ipsum an	Lorem Ipsan	Split	Lorem	Lor em
Reject	Lorem Ipsum	Lorem	Validate	Lorem Ipsum	Lorem Ipsum
Merge	Lorem Ipsum	LoremIpsum	Undo	Lorem	Lorem Ipsum
Delete	Lorem Ipsum	Lorem	Redo	Lorem Ipsum	Lorem
Insert	Lorem Ipsum et	Lorem et Ipsum	Help	Lorem Ipsum	<help event>

Figure 10: Set of gestures

2.2 HTR server

The 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.

3 T5.5 Machine Translation Server

In the CASMACAT workbench, the UI (implemented in Javascript running in a web browser) connects to the Computer Aided Translation (CAT) server via web sockets. The CAT server implements interactive translation prediction as described in Section 1.1.3, alongside other advanced types of assistance such as confidence measures and word alignment. 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.

We separate the functionalities of the MT server and the CAT server to support more modularity and allow for the use of multiple MT server implementations, and even external MT services. Within the CASMACAT, we bring together two machine translation systems, Thot and Moses, and extend them to support the requirements of the project. In this section we describe how these machine translation systems are accessible as a server process that response to API calls via TCP/IP.

3.1 Translation

The main call to the server is a request for a translation. In the request, the source sentence (`q`), source and target language (`source`, `target`) and optionally a key identifying the user (`key`). Here is an example request:

```
http://demo.casmacat.eu:8000/translate?q=test&key=0&source=en&target=es
```

The server responds to requests with an JSON object.

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

Note that this is the same API specification as Google Translate. Our server implementation extends this API in various ways.

3.2 Report tokenization and alignment

Tokenization is reported by default. Now also the alignment should make sense since this is based on the tokenized and preprocessed src and raw output. Alignment info is returned including a parameter `align` in the request. The alignment can deal with cases like:

- raw: `Mein Haus @-@ Tier , &gt; Fisch .`
- postprocessed: `Mein Haus-Tier , > Fisch.`

from which these spans would be extracted:

- raw:
  `[[0, 3], [5, 8], [10, 12], [14, 17], [19, 19], [21, 24], [26, 30], [32, 32]]`
- postprocessed:
  `[(0, 3), (5, 8), (9, 9), (10, 13), (15, 15), (17, 17), (19, 23), (24, 24)]`

representing these tokens:

- raw: `| Mein | Haus | @-@ | Tier | , | &gt; | Fisch | . |`
- postprocessed: `| Mein | Haus | - | Tier | , | > | Fisch | .`

3.3 N-Best Lists

By adding `nbest=n` to the request the server gives an n-best list of size n in two formats, raw and post-processed. The latter format consists of a list of entries which look similar to what we get for the first-best translation the former contains the raw output and scores:

```
"raw_nbest":
  [{"hyp": "the American President Obama comes after Oslo . ",
    "totalScore": -21.49662780761718
    },
    {"hyp": "the American President Obama comes after Oslo . &quot; ",
     "totalScore": -23.7340850830078
    },
    [...]
  ]
```

3.4 Search Graphs

By adding `sg` to the request the server also returns a search graph. The search graph is a set of states and transitions that mirror the process of hypothesis generation during decoding. The search graph currently uses the Moses format for search graphs. See below for an example for the format.

```
"searchGraph":
  [{"forward": 1.0,
    "hyp": 0,
    "stack": 0,
    "fscore": -2.447231531143188
   },
   {"transition": -2.447231531143188,
    "back": 0,
    "hyp": 1,
    "score": -2.447231531143188,
    "cover-start": 0,
    "forward": -1.0,
    "cover-end": 0,
    "stack": 1,
    "fscore": 0.0,
    "out": "test"},
   {"transition": -2.85275387763977,
   [...]
   }
  ]
```

3.5 Word Posterior Probabilities as Word Level Confidence Estimate

Re-using the n-best list all entries are aligned to the first-best hypothesis and posterior probs are computed using the scores. Use 'wpp=n' to set length of the nbest list used to this purpose. The resulting values are just a single float per token.

Example (on toy data):

```
"translatedText": "The American President Obama comes after Oslo.",
"wpp": [1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 0.99 ],
```

4 T5.6 Manual Gaze-to-Word Alignment

The DOW specifies that "We will implement a tool for aligning and correcting erroneous fixations so as to manually map them on the words that are likely to be fixated. The tool will be instrumental in creating a corpus of high-quality gaze-to-word mappings, and will be used as a training set of automatic gaze-to-word mappings algorithms."

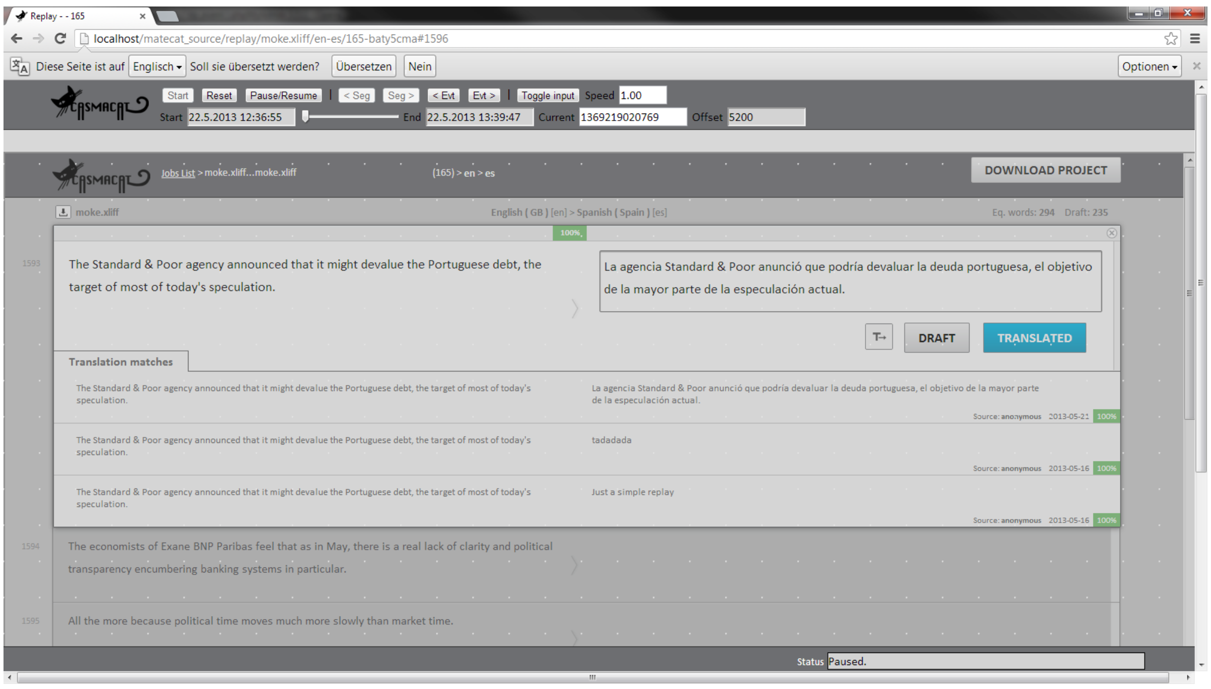
A first prototype of the manual alignment tool has been implemented in Translog-II and fixations in 64 files were manually re-mapped. These files are part of TPR-DB v1.0 and can be downloaded through the TPR-DVv1.0 svn server. The re-mapped files include all 10 translation sessions in the study BD08 and 54 files of the study BML12.

5 T5.7 Automatic Gaze-to-Word Alignment

We implemented two algorithms for automatic gaze-to-word alignment. The manually re-mapped fixations were used to evaluate the precision of these algorithms. The algorithms and their evaluation are described in two publications (one published, one draft), which are attached in the appendix of this deliverable.

- Abhijit Mishra, Michael Carl and and Pushpak Bhattacharyyya (2012). A heuristic-based approach for systematic error correction of gaze data for reading. Proceedings of the First Workshop on Eye-tracking and Natural Language Processing, December 2012, Mumbai, India (`http://www.aclweb.org/anthology/W/W12/W12-4906.pdf`)
- Dynamic programming for re-mapping noisy fixations in translation tasks. Draft.

We are currently working on a comparison of the two programs.


Replay - - 165
localhost/matecat_source/replay/moke.xliff/en-es/165-baty5cma#1596
Diese Seite ist auf Englisch Soll sie übersetzt werden? Übersetzen Nein Optionen
Start Reset Pause/Resume < Seg Seg > < Evt Evt > Toggle input
DOWNLOAD PROJECT
DRAFT
TRANSLATED
Translation matches

Speed 1.00
Start 22.5.2013 12:36:55 End 22.5.2013 13:39:47 Current 1369219020769 Offset 5200

Jobs List > moke.xliff...moke.xliff (165) > en > es

moke.xliff English (GB) [en] > Spanish (Spain) [es] Eq. words: 294 Draft: 235

100%

1593 The Standard & Poor agency announced that it might devalue the Portuguese debt, the target of most of today's speculation.

La agencia Standard & Poor anunció que podría devaluar la deuda portuguesa, el objetivo de la mayor parte de la especulación actual.

The Standard & Poor agency announced that it might devalue the Portuguese debt, the target of most of today's speculation.	La agencia Standard & Poor anunció que podría devaluar la deuda portuguesa, el objetivo de la mayor parte de la especulación actual.	Source: anonymous 2013-05-21 100%
The Standard & Poor agency announced that it might devalue the Portuguese debt, the target of most of today's speculation.	tadadada	Source: anonymous 2013-05-16 100%
The Standard & Poor agency announced that it might devalue the Portuguese debt, the target of most of today's speculation.	Just a simple replay	Source: anonymous 2013-05-16 100%

1594 The economists of Exane BNP Paribas feel that as in May, there is a real lack of clarity and political transparency encumbering banking systems in particular.

1595 All the more because political time moves much more slowly than market time.

Status Paused.

Figure 11: Replay view

6 T5.8 Replay Mode for User Activity Data

The replay view now works very differently from the one of the 1st prototype. The replay view loads the translate view into an *iframe* and remote-controls it with the data from the log file. This is more robust, requires less changes in the translate view and allows for changes of the replaying window geometry (like resizing).

The log data is now fetched in small chunks when replaying. When replay is started, 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. This was a problem in the 1st prototype where the whole log data has been fetched first before starting the replay, which took easily up to 15 minutes and more. The new loading mechanism may also be optimized in the future so that the next chunk(s) are pre-fetched, while replay is still running. In this way there will only be a short waiting time at the beginning of the replay (or, of course, if the user starts seeking).

The new replay engine now uses a more precise internal clocking. In the 1st prototype, events with the same timestamp have been grouped and replayed together. Now, each event is replayed on its on. This makes the engine more precise, more robust and allows for arbitrary jumps between events.

A lot of new events have been added to the logging (especially ITP) and those new events are not yet fully integrated in the replay mode. Currently, only PE session are replayed. But the functionality will be extended in the next weeks to include the new events and 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. See Figure 11 for a screenshot.

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

The replay of a document can be reached either by going to the view 'List Documents' (see 1.1.5) or by changing the string 'translate' to 'replay' in the URL of a document. For instance, the URL

```
http://localhost/matecat_source/translate/moke.xliff/en-es/171-zwrczyxk
```

would become

```
http://localhost/matecat_source/replay/moke.xliff/en-es/171-zwrczyxk
```

7 T5.9 Visualization of Translation Processes

Besides visualising of the translation process data in a replay mode, the Translation Process Research Database (TPR-DB) also allows for plotting translation sessions in the form of translation progression graphs. However, translation progression graphs only visualize a small fraction of the information that is contained in the TPR-DB.

A overview of the features that are contained in the database and their current visualization possibilities are describe a draft paper which is included in full in the appendix.

- Feature Representation in the Translation Process Research DB. Draft.

References

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

[2] Sergio Barrachina, 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.

[3] Peter F Brown, 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.

[4] Luis A. Leiva, 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.

[5] G. Sanchis-Trilles, Daniel Ortiz-Martínez, Jorge Civera, Francisco Casacuberta, Enrique Vidal, and Hieu Hoang. Improving interactive machine translation via mouse actions. In *Proc. EMNLP*, 2008.

Appendix

This appendix contains three publications that give more details to the automatic gaze-to-word alignment methods mentioned in Section 5 and visualization of translation processes in Section 7.

- Abhijit Mishra, Michael Carl and and Pushpak Bhattacharyyya (2012). A heuristic-based approach for systematic error correction of gaze data for reading. Proceedings of the First Workshop on Eye-tracking and Natural Language Processing, December 2012, Mumbai, India (`http://www.aclweb.org/anthology/W/W12/W12-4906.pdf`)
- Dynamic programming for re-mapping noisy fixations in translation tasks. Draft.
- Feature Representation in the Translation Process Research DB. Draft.

A heuristic-based approach for systematic error correction of gaze data for reading

Abhijit Mishra *Michael Carl* *Pushpak Bhattacharya*
(1) Indian Institute Of Technology, Bombay
(2)CRITT, Copenhagen Business School
(3) Indian Institute Of Technology, Bombay

abhijitmishra@cse.iitb.ac.in, mc.isv@cbs.dk, pb@cse.iitb.ac.in

ABSTRACT

In eye-tracking research, temporally constant deviations between users' intended gaze location and location captured by eye-samplers are referred to as systematic error. Systematic errors are frequent and add a lot of noise to the data. It also takes a lot of time and effort to manually correct such disparities. In this paper, we propose and validate a heuristic-based technique to reduce such errors associated with gaze fixations by shifting them to their true locations. This technique is exclusively applicable for reading tasks where the visual objects (characters) are placed on a grid in a sequential manner; which is often the case in psycholinguistic studies.

KEYWORDS: EYE-TRACKING, FIXATION CORRECTION, GAZE DATA MANIPULATION, SYSTEMATIC ERROR

1 Introduction

In psycholinguistic studies, eye tracking experiments have often been conducted to study the human way of analysing and synthesizing text. During reading, eye movement significantly relates to the cognitive load on participants. So, analysing gaze data is useful in proving/disproving hypotheses and extracting features for training and tuning machines. But eye trackers, after all, have certain limitations and they exhibit error in capturing gaze points of individuals. Such errors could be classified into variable and systematic errors (Harnof and Halverson, 2002). Variable error is nothing but dispersed gaze-points around the intended fixation which indicate lack of precision of eye-trackers. Systematic error, on the other hand, is the drift between the gaze-point locations captured by the eye-trackers and the intended fixation. It may be caused by imperfect calibration, head movement, astigmatism and other sources (LC Technologies, 2000). With the advent of sophisticated eye-trackers (Tobii, SR Research Eyelink etc.) it has been possible to reduce variable errors. But yet there is still a demand of tools and techniques to handle systematic errors which often imposes adverse impact on gaze-point analysis.

Various methods have been proposed to handle systematic error associated with fixations. Abrams and Jonides (1988) and Juhasz et.al (2006) proposed recalibration in the course of experiment which may not be applicable for linguistic analysis since such interruptions would reduce the quality of task. For example: during translation process studies participants cache contextual evidences in their short term memory, which could be lost by such interruptions.

Hornof and Halverson (2002) introduced Required Fixation Location (RFL) technique in which they identify RFLs i.e some points on the screen which indicates the actual fixation of the candidates at a specified time. In some of the experiments they record RFLs by asking participants to place the mouse cursor over the objects they were looking at. Then they measure the discrepancies between RFLs and fixations recorded by eye-trackers and shift the fixations to the true locations. This method is not very useful where one cannot ask the user to indicate RFLs. For example, during translation studies the participant might be busy typing the translations and reading the text simultaneously. Similar is the case with annotation tasks where the user has to read and annotate a text.

The Gaze to Word Mapping (GWM) modules introduced by Špakov, (2007) is a heuristic based approach. The underlying algorithm does not make a simplistic link between the x-y coordinates of a fixation and the location of a word on the monitor, but rather tries to account for certain documented effects, closely resembling to our technique. While is it quite reasonable to believe that participants tilt towards the end of reading lines; it doesn’t clearly show us a way to determine the line which the participant is looking at; given initial few fixations are nonlinear in nature. Our algorithm tries to overcome this by introducing a scoring function which guesses which line a participant is focusing on; given N initial non-linear/linear fixations starting at time T.

The Mode-of-disparities error correction technique proposed by Zhang and Hornof (2011) is useful when the visual objects are arranged in an irregular manner but fails when objects are placed on a grid such as placing a paragraph for reading.

Intuitively, for reading and writing tasks vertical displacement of fixations contribute more to the noise than that of horizontal. So in this article, we focus more on vertical directional adjustment.

Initially, before processing fixations, a set of virtual horizontal lines are drawn by joining the centre coordinates of character belonging to the respective textual lines. Fixations are extracted from the noisy data and stored sequentially in a temporal order[1]. Then they are processed and corrected in three stages. In first stage, fixations are shifted to lie on the nearest virtual lines. In the second stage transient fixations are corrected. Finally, participant's Reading Line (RL) is predicted and deviating fixations are shifted to the corresponding RLs.

This technique is applied on the Translation Process Research (TPR) database (Carl, 2012) recorded by Tobii eye-tracker using Translog-II (Carl 2012) software. Then validation is done across manually corrected fixations. Qualitative analysis is done by replaying the recorded and corrected data in Translog. In all the cases we have assumed left to right reading and writing but the technique could be slightly modified to support for languages adopting Arabic scripts.

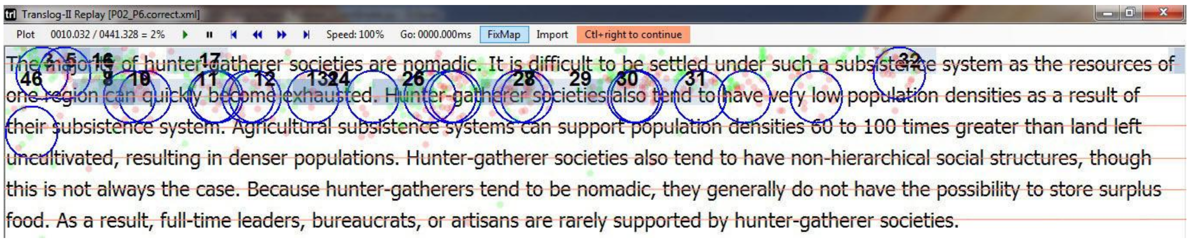
2 Heuristics for Fixation Correction

In order to hand code rules for fixation correction, we have extensively studied the fixation sequences in TPR database. The database contains more than 450 recordings for translation, post-editing and reading experiments in 7 languages and are collected over last 5 years by a following a systematic initial experimental setup (Carl, M. and Jakobsen A.L. 2009); the eye-tracker used being Tobii, a remote eye-tracker. However, this does not bias our heuristics since many of the psycholinguistic experiments involving reading and writing tend to follow similar set-up. Moreover, other state of the art remote eye-trackers (such as SR research, SMI vision) report same or more accuracy as Tobii.

Fixations in the recorded data are corrected in three phrases as described below.

2.1 Shifting fixations to the nearest line

First of all, recorded fixations could be dispersed over the screen whereas the intended fixation should only possibly lie on visual objects such as characters. A fixation lying on the blank space between two lines is nothing but an indication of error. So the first step is to shift the fixations vertically to the nearest line. To come up with discrete lines we have taken the cursor coordinates of each character in a line and joined them to draw a virtual line. Figure 1 illustrates a set of virtual lines going through the text. These lines serve as Reading Lines (RLs) in the later processing stages.



The majority of hunter-gatherer societies are nomadic. It is difficult to be settled under such a subsistence system as the resources of one region can quickly become exhausted. Hunter-gatherer societies also tend to have very low population densities as a result of their subsistence system. Agricultural subsistence systems can support population densities 60 to 100 times greater than land left uncultivated, resulting in denser populations. Hunter-gatherer societies also tend to have non-hierarchical social structures, though this is not always the case. Because hunter-gatherers tend to be nomadic, they generally do not have the possibility to store surplus food. As a result, full-time leaders, bureaucrats, or artisans are rarely supported by hunter-gatherer societies.

FIGURE 1 – Shifting fixations to nearest virtual lines

[1] Fixation sequencing is done on the basis of time of occurrence of the fixations. For exemple, if we say a particular fixation (say F2) follows/precedes another fixation (F1), we mean, F1 occurs sooner/later than F2 even if F2 appears to the left/right of F1 co-ordinate wise.

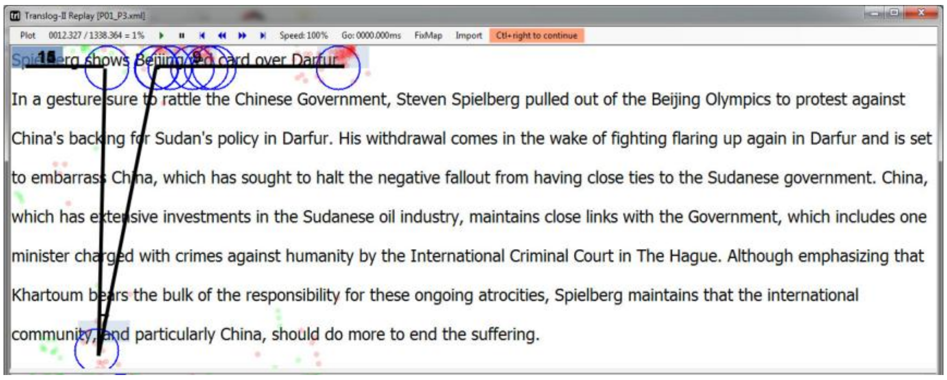
Figure 1 is a screen dump of Translog II, The orange lines represent virtual lines (Reading Lines). The red and green dots represent gaze samples of left and right eyes and blue circles represent fixations.

Sometimes, shifting fixations to the nearest virtual line is not enough. Upon closely looking at figure-1, one would predict that the participant is trying to read line1. But after shifting the fixations most of the fixations fall on line2.

After this step, it becomes easy to obtain systematic patterns which reduces the randomness and hence, the number of rules to be used for correction.

2.2 Discarding transient fixations

Transient Fixations (TFs) are very short duration fixations which occur in between two fixations falling nearer to each other (on the same line or just a line apart) and located far away from each of them. In other words, upon joining three fixations if we observe a spike and the tip of the spike is a short duration fixation, it is said to be transient. Figure 1 illustrates one TF.



Spielberg shows Beijing red card over Darfur

In a gesture sure to rattle the Chinese Government, Steven Spielberg pulled out of the Beijing Olympics to protest against China's backing for Sudan's policy in Darfur. His withdrawal comes in the wake of fighting flaring up again in Darfur and is set to embarrass China, which has sought to halt the negative fallout from having close ties to the Sudanese government. China, which has extensive investments in the Sudanese oil industry, maintains close links with the Government, which includes one minister charged with crimes against humanity by the International Criminal Court in The Hague. Although emphasizing that Khartoum bears the bulk of the responsibility for these ongoing atrocities, Spielberg maintains that the international community, and particularly China, should do more to end the suffering.

FIGURE 2 – Transient Fixations

Figure 2 shows one transient fixations. Upon joining 3 consecutive fixations involving one TF, we observe a spike.

In some studies, we do not need TFs to be present in our data as the fixation count un-necessarily grows on account of TFs. Transient fixation may also add noise to the data in some cases where, for example, fixation count for a region is a part of our study. Suppose, for our translation studies if we want to count fixations in source text window (src) and target text window (tgt) during an interval of 20 seconds and a lot of transient fixations fall on tgt, the distribution will be completely different from that of if we discard transient fixations. Such cases would require discarding TFs.

2.3 Correcting continuous abnormalities in fixation sequences

In this stage we try to predict the Reading Line (RL) of the participant at a specified time period and try to shift way-ward fixations within that time period to the corresponding RL. For instance, consider the case where the user starts reading the text from left to right and the eye-tracker records F fixations within the timespan of T. After shifting those fixations to the nearest lines, it is observed that first N out of the F fixations lie on line1. Here we can, to some extent, believe that the RL for the participant for the timespan T is line1. Now suppose the rest (F-N) fixations lie on line2 and the X co-ordinate of these fixations are greater than those of first N fixations. In this case, it is unlikely that the RL of the user has changed from line 1 to line2. Hence those (F-N) fixations have to be relocated to line1.

Assuming that the initial calibration is perfect enough for a particular experiment session and the line spacing width significant (which is often the set up in linguistic studies) , it is reasonable to believe that most of the first N (co-ordinate wise) fixations decide the RLs. The intuition behind such an assumption is that, if the participant is reading from left to right, after reading certain words from left, there will be a gradual head movement and tilting which might contribute to shifting of fixations to the next/previous line.

The value of N is decided by taking samples from the recorded data and observing it by replaying the recordings. It is highly possible that the first N fixations could be distributed amongst different lines; each being a candidate RL. In such cases we infer the RL by ranking the candidates as follows

$$RL = \operatorname{argmax}_{r\in R} \sum_{f\in firstN} \sum_{r\in R} (\delta_r (f.Y) \times dur(f))$$

where R is the set of RLs, δ is Dirac Delta function and dur(f) is duration of fixation f

The first part of the summation represents fixation frequency distribution amongst the RIs. The intuition behind taking such a function is that during reading/writing, fixation duration and frequency are measurable factors providing evidences regarding participant's attention. The rationale behind taking Dirac Delta is that one particular fixation at time T could lie only on one Reading Line.

If the scores of two potential RLs match, RL is assigned to the line having maximum fixation. If that still matches, random assignment has to be done. Once the RL for a particular time period has been detected, the following two types of deviations are corrected.

Type A: This is a case when the user tries to read Mth line from left to right. A few fixations (say P) lie on line M spatially followed by a number of fixations (say F) on line M+1. The x-coordinates of those F fixations are greater than those of P. In such cases those F fixations are shifted upward to line M unchanging x-coordinates. (Figure 3 Type A)

Type B: Here, the user tries to read Mth line from left to right. A few fixations (say P) lie on line M spatially followed by a number of fixations (say F) on line M-1. The x-coordinates of those F fixations are greater than those of P. In such cases those F fixations are shifted downward to line M unchanging x-coordinates. (Figure 3 Type B)

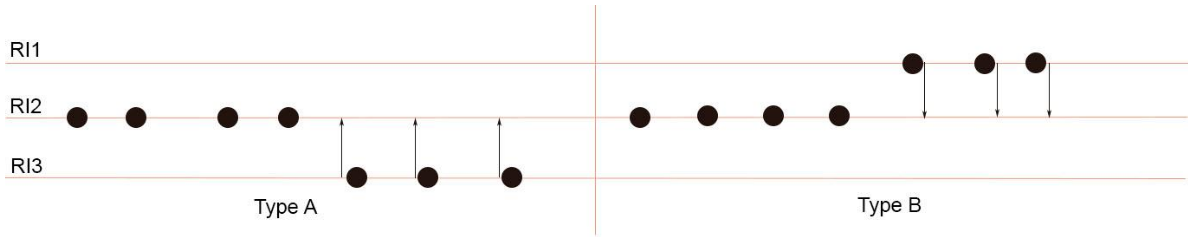


FIGURE 3 – Type A and Type B deviations

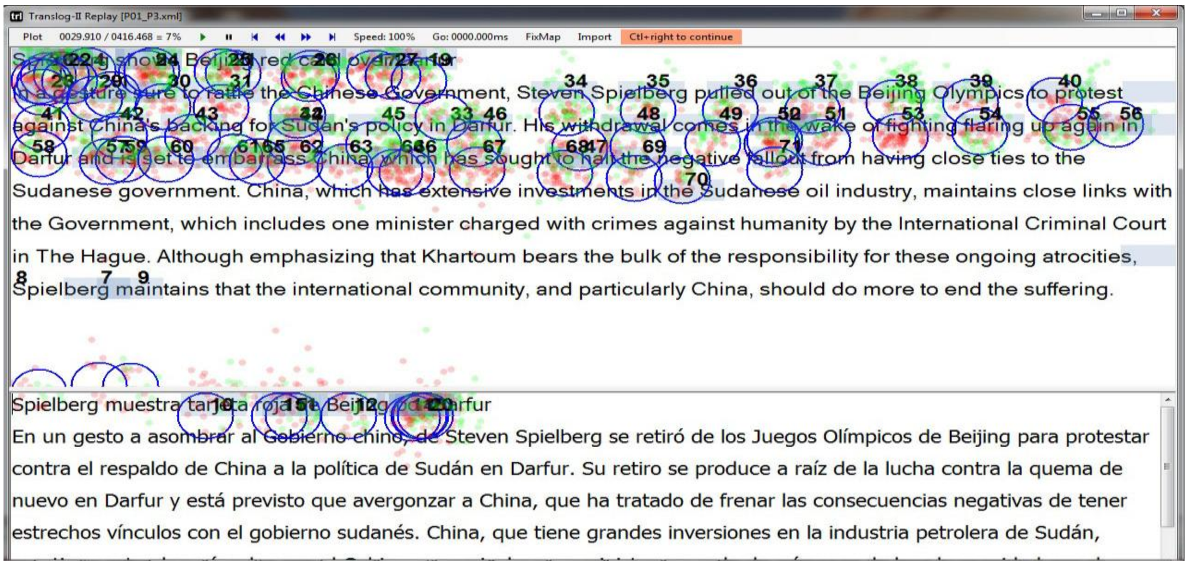
3 Algorithm

```
correctFixations (N, loggedData):
    fixationSet := extractFixations(loggedData)
    fixationSet = sortByTimeOfOccurrence (fixationSet)
    RL_Set := extractDistinctYCoordinate(loggedData)
    Foreach fixation in fixationSet:
        Re-assign the y-coordinate of the fixation to that of the closest RL
    correctTransientFixations (fixationSet)
    correctAbnormalities (fixationSet,N,RL_Set)
    Update logged data with fixationSet
    return
correctTransientFixations (fixationSet):
    averageFixationDuration := ComputeAvarageFixationDuration(fixationSet)
    Foreach fixation in fixationSet:
    IF previousFixation doesn’t exist OR nextFixation doesn’t exist
        Continue
    IF abs(previousFixation.Y-nextFixation.Y) << abs(previousFixation.Y -fixation.Y)
            AND fixation.duration  << averageFixationDuration)
        Delete fixation from fixationSet
correctAbnormalities (fixationSet,N,RL_Set):
    startingPoint := 1
    firstN: = selectNFixations(fixationSet, startingPoint,N)
    RL:= getRLWithMaximumScore(firstN,RL_Set)
    X: = getLargestXCoordinate(firstN,RL)
    targetSet: = setDifference(fixationSet,firstN)
    Foreach fixation in targetSet -:
        startingPoint+=1
        L1 = getLineNumber(fixation.Y)
        L2 = getLineNumber (RL)
        IF previousFixation doesn’t exist OR nextFixation doesn’t exist
            Continue
        IF (previousFixation.X > fixation.X and previousFixation.X>nextFixation.X)
            RL = getRLWithMaximumScore(firstN,RL_Set)
            X = getLargestXCoordinate(firstN,RL)
            targetSet = setDifference(fixationSet,firstN)
            Continue
        IF (abs(L2-L1)==1 and fixation.X >X)
            fixation.Y = RL
getRLWithMaximumScore (firstN,RL_Set)
    RL = $\text{argmax}_{r\in RI\_Set} \sum_{f\in firstN} \sum_{r\in RI\_Set} \delta_r (f.Y) \times dur(f)$
    Return RL
```

The subroutines selectNFixations returns N fixations from the starting index. Similarly, getLargestXCoordinate returns the right-most fixation lying on an RL.

4 Validation

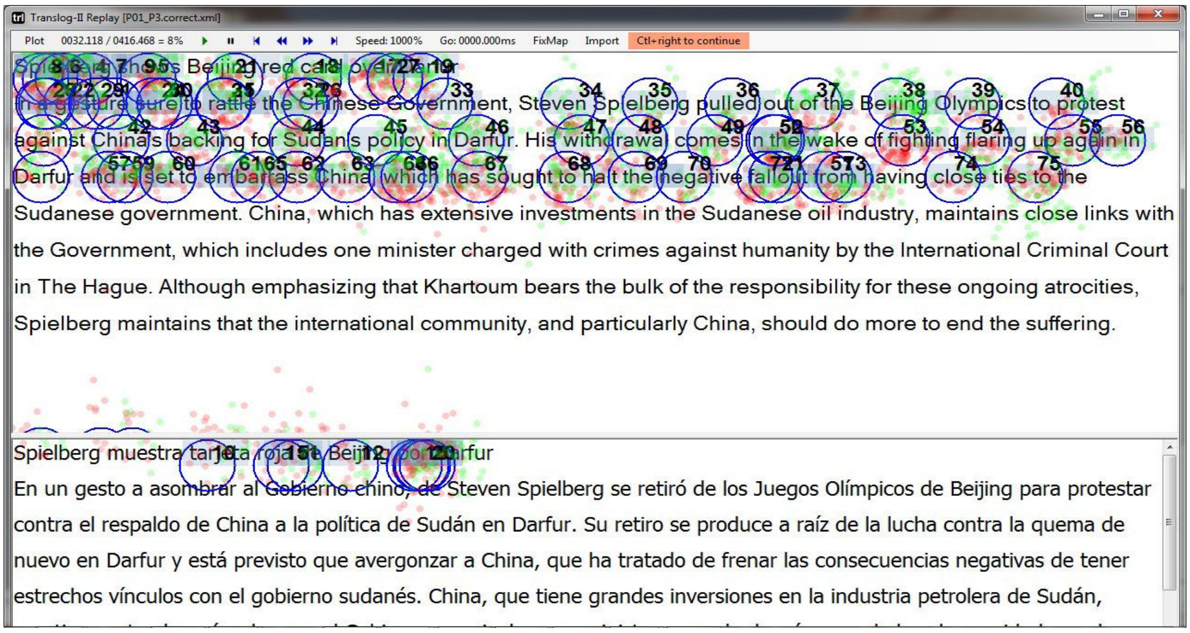
This technique was applied on Spanish and Danish translation and post-editing recording sessions from Translation Process Research (TPR) database. Qualitative analysis of the corrected data showed improvement.


Translog-II Replay [P01_P3.xml]
Plot 0029.910 / 0416.468 = 7%
Speed: 100%
Go: 0000.000ms
FixMap
Import
Ctl+right to continue

Spielberg shows Beijing red card over Darfur
In a gesture sure to rattle the Chinese Government, Steven Spielberg pulled out of the Beijing Olympics to protest
against China's backing for Sudan's policy in Darfur. His withdrawal comes in the wake of fighting flaring up again in
Darfur and is set to embarrass China, which has sought to halt the negative fallout from having close ties to the
Sudanese government. China, which has extensive investments in the Sudanese oil industry, maintains close links with
the Government, which includes one minister charged with crimes against humanity by the International Criminal Court
in The Hague. Although emphasizing that Khartoum bears the bulk of the responsibility for these ongoing atrocities,
Spielberg maintains that the international community, and particularly China, should do more to end the suffering.

Spielberg muestra tarjeta roja a Beijing por Darfur
En un gesto a asombrar al Gobierno chino, de Steven Spielberg se retiró de los Juegos Olímpicos de Beijing para protestar
contra el respaldo de China a la política de Sudán en Darfur. Su retiro se produce a raíz de la lucha contra la quema de
nuevo en Darfur y está previsto que avergonzar a China, que ha tratado de frenar las consecuencias negativas de tener
estrechos vínculos con el gobierno sudanés. China, que tiene grandes inversiones en la industria petrolera de Sudán,

FIGURE 5 – Uncorrected fixations


Translog-II Replay [P01_P3.correct.xml]
Plot 0032.118 / 0416.468 = 8%
Speed: 1000%
Go: 0000.000ms
FixMap
Import
Ctl+right to continue

Spielberg shows Beijing red card over Darfur
In a gesture sure to rattle the Chinese Government, Steven Spielberg pulled out of the Beijing Olympics to protest
against China's backing for Sudan's policy in Darfur. His withdrawal comes in the wake of fighting flaring up again in
Darfur and is set to embarrass China, which has sought to halt the negative fallout from having close ties to the
Sudanese government. China, which has extensive investments in the Sudanese oil industry, maintains close links with
the Government, which includes one minister charged with crimes against humanity by the International Criminal Court
in The Hague. Although emphasizing that Khartoum bears the bulk of the responsibility for these ongoing atrocities,
Spielberg maintains that the international community, and particularly China, should do more to end the suffering.

Spielberg muestra tarjeta roja a Beijing por Darfur
En un gesto a asombrar al Gobierno chino, de Steven Spielberg se retiró de los Juegos Olímpicos de Beijing para protestar
contra el respaldo de China a la política de Sudán en Darfur. Su retiro se produce a raíz de la lucha contra la quema de
nuevo en Darfur y está previsto que avergonzar a China, que ha tratado de frenar las consecuencias negativas de tener
estrechos vínculos con el gobierno sudanés. China, que tiene grandes inversiones en la industria petrolera de Sudán,

FIGURE 6 – Automatically corrected fixations

As we can see in the initial data (Figure 5), the fixation distribution is noisy and there is an overlap among fixations lying on line 3 and 4. After correction (Figure 6) the noise is significantly reduced. Fixations are labelled as per their temporal ordering.

5 Comparison with manual correction

We compared our output with manual corrections done for Spanish and Danish TPR data. Since our method shifts most of the fixations and manual correction only involves correcting only certain badly shifted fixations by mapping an appropriate word to the fixation, we checked for what fraction of manual correction could be successfully carried out by our method.

First, we mapped our fixations to the words on which they lie. Then from the original data we took the timestamp of those fixations which were corrected manually. For those timestamps we collected Fixation-to-word mapping for both the corrected versions and produced the Longest Overlapping Subsequence (LOS) between the mapped words. If the length of the LOS is more than the sum of the character counts of those two corresponding words, it is considered to be a valid correction.

For different values of N, we checked for the percentage of correction done with respect to manual correction. The results are shown by the following table

	N=3	N=6	N=10
Danish (10 sessions)	63%	83%	79%
Spanish (40 sessions)	55%	81%	81%

TABLE 1 – Automatic Vs Manual Correction

6 Conclusions

In this article, we presented a mechanism to correct systematic error associated with fixations by applying certain heuristics. The advantage of this method is, it can be applied both online (in the course of experiments) and offline. But the correction quality depends on the value of N and other parameters like initial experimental set-up and degree of randomness of fixations etc. It works best for shallow visualization studies; making it quite useful in studies like Translation Process Study, Sentiment Analysis etc.

There are certainly several factors for drift and imprecision apart from what we have taken into account. For instance, if the eye-tracker maps all gaze sampled, say 3cm below the intended location (because the head was permanently moved), all gaze samples are 3cm distorted, including the ones on the first N words in a line. Our algorithm fails to detect this. Of course, for the studies involving writing, we can get this constant drift (3cm) by comparing the cursor and the fixation positions during writing and finding out the average deviations. This is somewhat similar to RFL techniques assuming that a person’s region of interest should not be very far away from the cursor position.

Our technique also fails if fixations are highly randomly distributed; which might be a case for studies involving detailed reading. In such cases, we also do not know the all the causes of the deviating fixations. Future work includes exploring and involving other case than just the two types of deviations that we took into account here. More cases and heuristics have to be included. A better validation technique has to be introduced as well.

References

Abrams, R. A., & Jonides, J. (1988). *Programming saccadic eye movements. Journal of Experimental Psychology: Human Perception and Performance*, 14, 428–443.

Hornof, A. J., & Halverson, T. (2002). *Cleaning up systematic error in eye-tracking data by using required fixation locations. Behavior Research Methods, Instruments, & Computers*, 34, 592–604.

Zhang, Y., & Hornof, A. J. (2011). *Mode-of-disparities error correction of eye-tracking data. Behavior Research Methods*, 43, 834–842. doi:10.3758/s13428-011-0073-0

Technologies, L. C. (2000). *The Eyegaze Development System: A tool for eyetracking applications. Fairfax,* VA

Carl, Michael (2012). Translog-II*: A Program for Recording User Activity Data for Empirical Reading and Writing Research*, In Proceedings of the *Eight International Conference on Language Resources and Evaluation, European Language Resources Association* (ELRA)

Carl, M. and Jakobsen A.L. (2009). *Towards statistical modelling of translators' activity data.* International Journal of Speech Technology, 12(4). http://www.springerlink.com/content/3745875x22883306/.

Carl Michael (2012). The CRITT TPR-DB 1.0: *A Database for Empirical Human Translation Process Research.* AMTA 2012 Workshop on *Post-Editing Technology and Practice* (WPTP-2012)

Špakov, O. (2007). GWM – *the Gaze-to-Word Mapping Tool*, available online at http://www.cs.uta.fi/~oleg/gwm.html.

Dynamic programming for re-mapping noisy fixations in translation tasks

Eyetracker which allow for free head movements are in many cases imprecise to the extent that reading patterns become heavily distorted. The reduced usability of these patterns for the gaze analysis is due to a "naïve" gaze-to-symbol mapping approach, which often wrongly maps the possibly drifted center of the observed fixation on the symbol directly below it. In this paper I extend this naïve fixation-to-word mapping by introducing background knowledge about the gazing task. In a first step, the sequence of naïve fixation-to-word mappings is projected into a lattice of several possible fixation locations, including those on the line above and below the naïve fixation mapping. In a second step a dynamic programming algorithm applies a number of heuristics to find the best path through the lattice, based on the likely distance in characters, in words and in pixels between successive fixations, so as to smooth the gazing path.

Keywords: Fixation-to-word mapping, drift in gaze data, drift-correction algorithm

Introduction

Translation Process Research has advanced to a state where recordings of behavioural data is used to elicit and model cognitive processes in the translators mind. In particular, the interrelation between the rhythm and speed of typing activities and the gazing behaviour is a valuable resource to understanding the translators black box. While the gazing behaviour reveals details about the text comprehension process, the typing of the translation shows us how the target text is produced and revised. In between these two activities lies the human translation process which we aim at understanding and modelling by looking at the physically measurable in- and output. The accuracy of the gaze data is crucial to obtain an undistorted approximation of these cognitive processes.

However, gaze data collected from eyetrackers is often noisy. The measured gaze location often does not exactly correspond to the spot that a subject actually looked at so that an analysis of the data may lead to misleading conclusions. This is harmful when studying gaze data during reading (or writing) activities where we deal with relatively small spacial areas - words or characters - on the screen that are read. A horizontal displacement of a few characters is still tolerable as it may still map to the same, or at least a neighbouring word, while a vertical displacement of only one line corresponds to a jump of perhaps 10 words, which may imply completely misleading conclusions when analysing the data. A vertical drift thus contributes more noise and may falsify major parts of the findings.

Noise and drift in gaze data has been addressed in several instances. A frequent method is to assess the collected data after an experimental sessions and disregard data which is too noisy. However, this seems unpractical in a setting which allows for free-head movements and which potentially add noise in almost every recording. Other methods make use of re-calibration on the fly Juhasz et.al (2006), or by means of Required Fixation Location (Hornof and Halverson, 2002). In this latter method, participants are asked to place the mouse cursor over the objects they are looking at. The measured discrepancies between the mouse cursor and the recorded fixations indicates a drift or noise offset between which may then be corrected. Also such methods are undesirable as they distract a translators, readers or writers form their usual working habit. Other solutions are necessary, in order to allow as much as possible for an ecologically valid working environment.

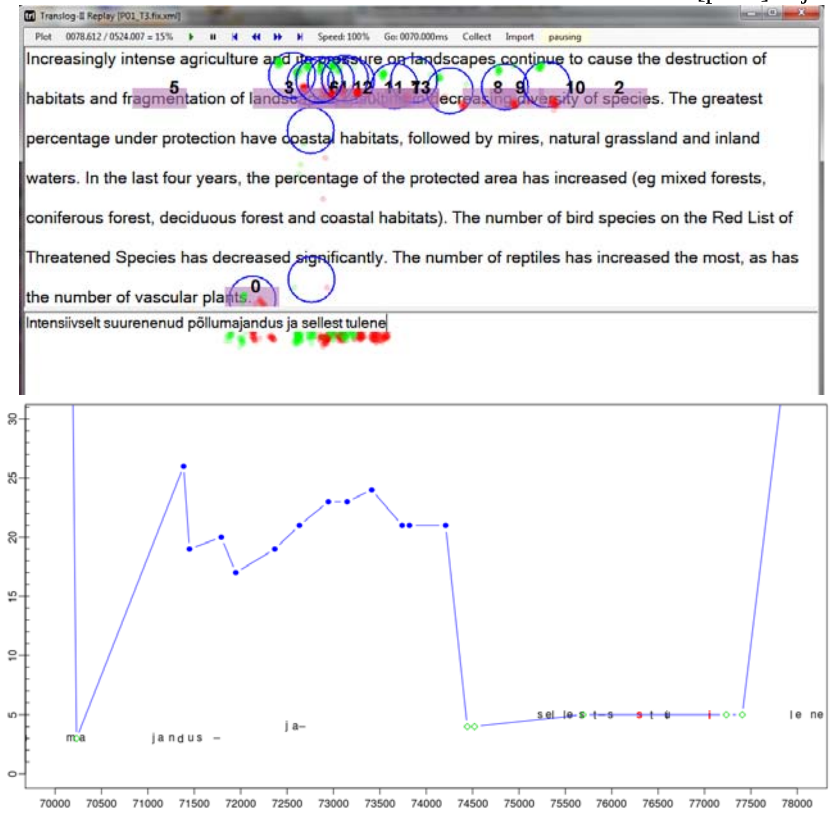
Mishra et al (2012) propose a heuristics-based technique to reduce temporally constant deviations between users' intended gaze location and the location captured by eye-samplers, so-called systematic errors. These error-correcting heuristics intend to shift gaze fixations to their "true locations", under the assumption that the measured gaze data at the beginning of a new line is often correct, while effects of gaze drift worsen as the eyes move towards the end of the line. Accordingly, the method of Mishra et al. places most importance to the first few fixations on each line, to which the remaining fixations are subsequently shifted. A similar method has been described by Špakov (2007), however, with a less sophisticated mechanism to determine the reading line from the first observed fixations.

In this paper we describe a fixation re-mapping algorithm that is tailored especially towards translation activities. In translation, the eyes move frequently between two texts, the source text and its translation, which calls for specific solutions. In the first section I discuss drifting problems in a sequence of recorded translation activity data, and why these drifting problems are difficult to capture with previous methods. From the description of the drifting problem I subsequently elaborate criteria for enhanced fixation-to-word mapping in translation tasks. The following section describes the implementation of the fixation remapping algorithm algorithm and a final section discusses evaluation issues.

The Problem

Figure 1 (top) shows an 8 second long fragment of a translation session from an English text into Estonian. In these 8 seconds were typed the characters "[pöllu]majandus ja sellest tulene" (excluding the part in square brackets) which - according to my back-translation from Estonian using google - corresponds to a translation of "agriculture and its pressure" in the English text.

The figure has three different types of gaze information: First: red dots represent gaze samples collected from the left eye, green dots are gaze samples of the right eye, and, second the blue circles represent fixations (i.e. clusters of coherent gaze points). The numbers on the fixations reflect their temporal ordering, so that fixation 0 occurred first, followed by fixation 2, then 3 etc. The third type of information are fixation-to-word mappings indicated by the violet background behind sequences of characters. Figure 1 shows a naïve fixation-to-word mapping. That is, the center of a fixation is mapped to the closest character and the background of the surrounding 8 characters are coloured in violet. These characters and words are then supposed to represent the words that were looked at by the



Increasingly intense agriculture and its pressure on landscapes continue to cause the destruction of habitats and fragmentation of landscapes resulting in decreasing diversity of species. The greatest percentage under protection have coastal habitats, followed by mires, natural grassland and inland waters. In the last four years, the percentage of the protected area has increased (eg mixed forests, coniferous forest, deciduous forest and coastal habitats). The number of bird species on the Red List of Threatened Species has decreased significantly. The number of reptiles has increased the most, as has the number of vascular plants.

Intensiivselt suurenenud põllumajandus ja sellest tulene

Figur 1 Replay situation with naïve fixation-to-word mapping showing a translation segment of 8 seconds. Top: the Translog-II replay shows the gaze sample points (red and green), the fixations and fixation to word mapping. Bottom: the translation progression graph shows the same segment of time with fixations on the source text (blue) fixations on the target text (green) and keystrokes on the time scale.

translator and are the basis of further analysis of reading behaviour.

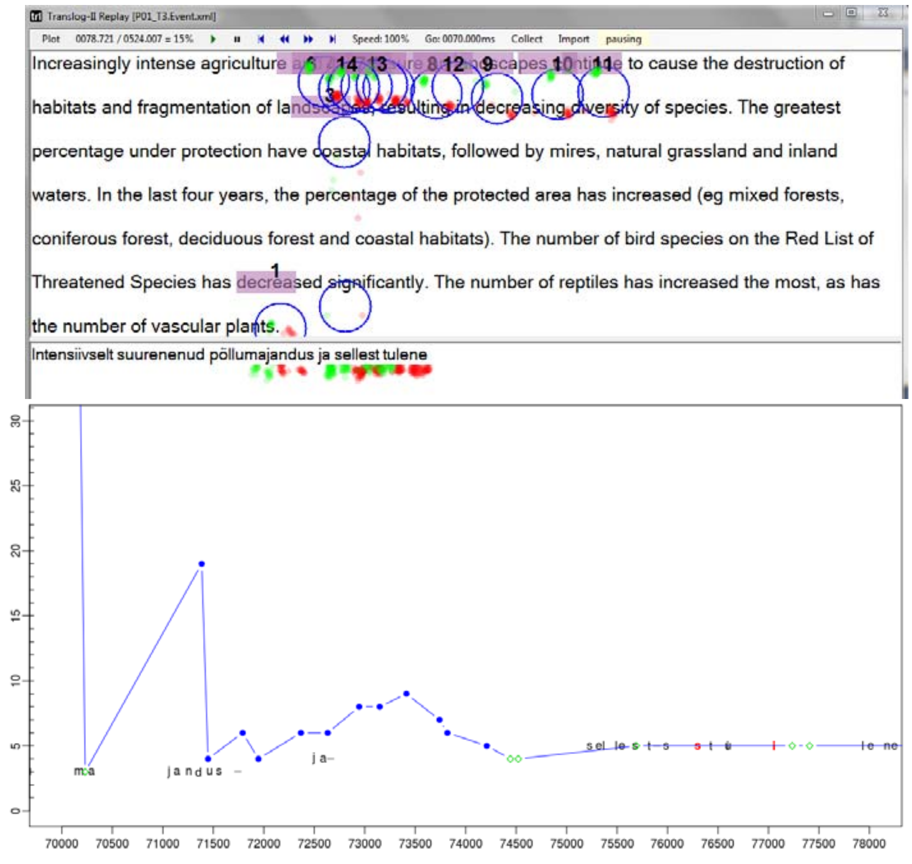
A number of imprecisions may distort this fixation-to-word mapping process, among others:

- due to calibration difficulties, free head movement or changing light or other conditions, the measured gaze sample points may not exactly correspond to the place which was gazed at.
- the choices that are taken when computing the fixation, e.g. based on the left or the right eye gaze sample, their average, how proximity or saccades between successive gaze samples are defined etc.
- the computation of the closest character for a given x/y position depends on which place of the character is taken as a reference, e.g. the upper left corner, or the center of the character, etc.

In Figure 1 (top), the fixations (blue circles) were computed based on the average of the left and the right eye sample, with the assumption that fixations should be at least 40ms in duration, and that all gaze samples within a fixation are no more than 25 pixels from the fixation center. In Figure 1 (top) most of the gaze samples lie between the first and the second line, but the fixations centers are mostly mapped on the words in the second line.

However, it is likely that the translator actually read a segment in the first line, since s/he is currently producing the translation of ”agriculture and its pressure” while the gaze moves back and forth between the source segment in the upper window and its translation in the lower window.

Figure 1 (bottom) shows the same segment in the form of a translation progression graph. The horizontal axis represents 8 seconds in which the fragment of the translation was typed (70.000ms to 78.000ms) while the vertical axis presents the source text to which the translation activities relate. The graph plots how the characters were typed in time: black characters are insertions and red characters deletions. The graph shows that there are several stretches of fluent writing (e.g. “jandus” and “ja”) and several pauses of different length (e.g. there is a pause of almost 1 sec between the typing of “jandus” and “ja”). Blue dots represent fixations on the source text words in the upper part of the Translog-II window while the green diamonds represent fixations on the translations in the lower window. Note that the blue dots in the bottom part in figures 1 and 2 correspond to the violet


Translog-II Replay [P01_T3.Event.xml]

Plot 0078.721 / 0524.007 = 15% Speed: 100% Go: 0070.000ms Collect Import pausing

Increasingly intense agriculture and its pressure on landscapes continue to cause the destruction of habitats and fragmentation of landscapes, resulting in decreasing diversity of species. The greatest percentage under protection have coastal habitats, followed by mires, natural grassland and inland waters. In the last four years, the percentage of the protected area has increased (eg mixed forests, coniferous forest, deciduous forest and coastal habitats). The number of bird species on the Red List of Threatened Species has decreased significantly. The number of reptiles has increased the most, as has the number of vascular plants.

Intensiivselt suurenenud põllumajandus ja sellest tulene

Figur 2: Replay situation with re-computed fixation-to-word mapping showing the translation segment of 8 seconds from figure 1. top: the Translog-II replay shows the gaze sample points (red and green), the fixations and fixation to word mapping a lineabove the mappings in figure 1. bottom: the translation progression graph shows the same segment of time with fixations on the source text (blue) fixations on the target text (green) and keystrokes on the time scale.

fixation-to-word mappings in the top part of the figure.

The segment shows that the translator was typing “ma” while the gaze was on the target window. The gaze moved then to the source window (blue dots), while typing “jandus ja” and then came back to the target, inspecting the just typed words (green diamonds), and then keeps on typing in “sellest tulene”, while correction a few typos (characters in red).

Figure 1 clearly shows the drift of gaze data and wrong fixation-to-word mapping in the source window: while the translation of source words 3 to 5 were typed, fixations in the source text are around words 17 to 26. However, it is likely that a translator read approximately the same words that s/he is currently translating, and not 12 or 20 words ahead.

A rectification of the fixation-to-word mapping to the first line in the source window is shown in Figures 2. While the location of the keystrokes, as well as the gaze samples and the computation of the fixation center are identical in the two pairs of figures, only the fixation-to-word mapping has changed. Figure 2 (bottom) shows the progression graph of the re-mapped segment, so that the distances between successive fixations become smoothed.

Even though we cannot be sure what a translator actually looked at – e.g. whether s/he read a segment in the first or second line - intuitively is seems more plausible that a translator reads source words which he or she is currently translating (as in Figure 2) instead of those words one line below (as in Figure 1). These observations lead us to criteria for a fixation-to-word re-mapping algorithm:

- successive fixations are more likely on neighbouring words than in the lines above or below
- translators are likely to read passages of source text which they are currently translating
- the distance between the fixation center and the fixated characters should be minimal

A Fixation-remapping Algorithm

Before applying the actual fixation re-mapping algorithm, we have re-computed fixations in a consistent manner with the following parameters:

- the minimum fixation duration was set to 40 ms
- each gaze-sample point must occur within 25 pixels from the center of the fixation
- a gap of gaze-sample data for more than 30ms would trigger a fixation boundary

The closest character to the median gaze sample within each fixation would then be taken as the fixation-to-word mapping. Figure 1 and Figure 4 show the results of this naïve fixation-to-word mapping, which was subsequently re-mapped based on the following algorithm.

In a first step, the sequence of "naïve" gaze-to-symbol mappings (as in Figure 1) is projected into a lattice of several possible gaze locations above and below the current fixation on the text. In a second step a dynamic programming algorithm applies a number of heuristics to find the best path through the lattice, based on the likely distance in characters, in words and in pixels between successive fixations, so as to smooth the gazing path according to observations reported in the literature.

Figures 3a to 3d illustrate this process based on the sequence of fixations and keystrokes between time stamps 70.000 and 73.000 in the previous figures 1 and 2. It illustrates the re-mapping of the fixation path in figure 1 (bottom) on the path plotted in figure 2 (bottom). Additional fixations are computed from the gaze sample points, in the following way:

- compute the fixation center only from the left eye gaze samples
- compute the fixation center only from the right eye gaze samples
- compute the fixation center from the average of the left and the right eye gaze samples

The fixation centers are then mapped on the closest nearby character in the source or target window, a so-called fixation-to-word mapping. There are thus three different fixation-to-word mappings `average`, `left` and `right`, depending on which fixation they are based on. In addition, a character is retrieved in the line above the upper most fixation-to-word mapping (`up`), and a character is retrieved in the line below the lowest fixation-to-word mapping (`down`). In this way, five fixation-to-word mappings are generated in addition to

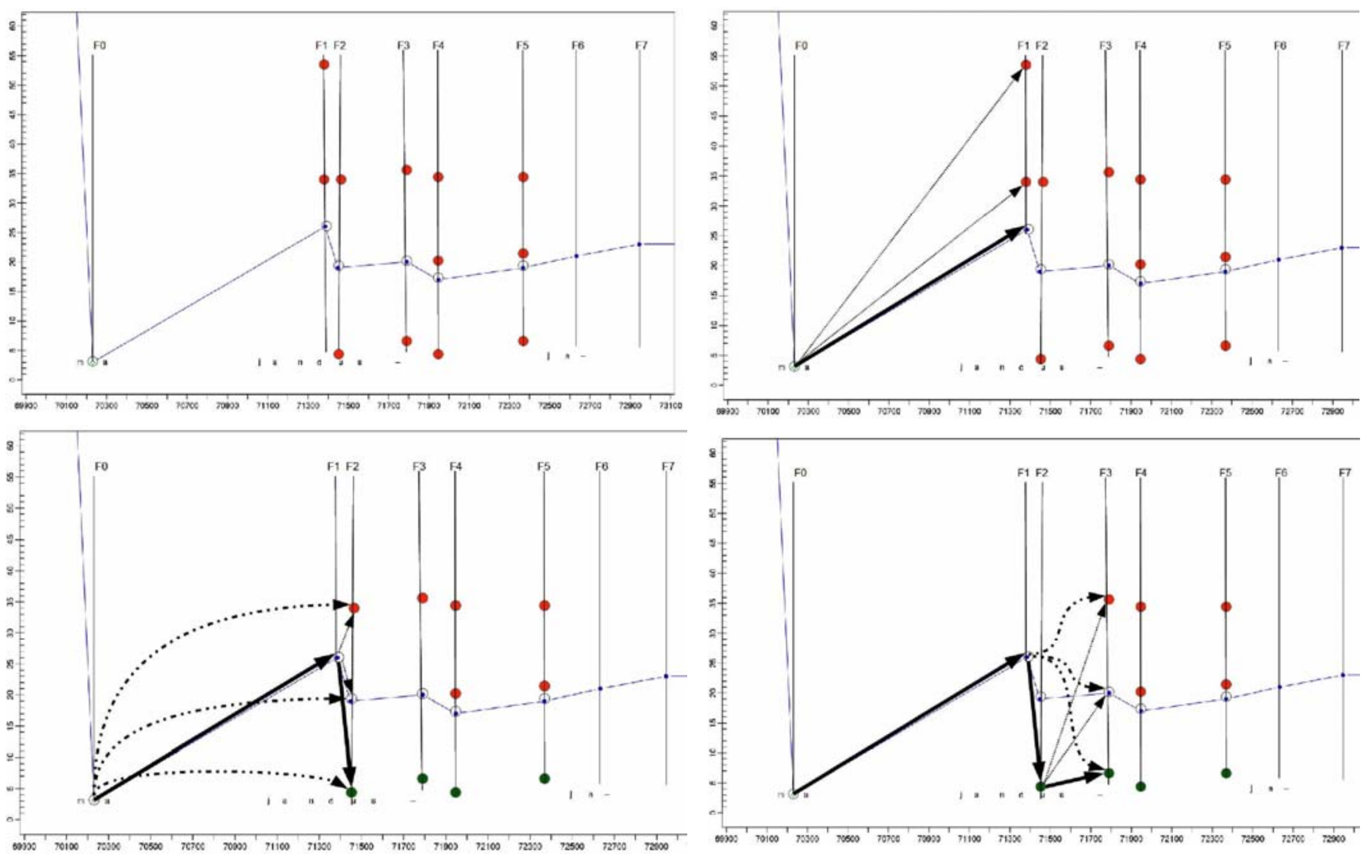


Figure 3a to 3d from top left to bottom right: Figure 3a shows the projection of the naïve mapping into a lattice of alternative fixation-to-word re-mappings (red dots). Figure 3b (top, right) plots links to the first successor node, figures 3c (bottom, left) and 3d (bottom, right) show successive steps in the re-mapping algorithm, including links to pre-predecessor nodes.

the original one, which may be however partially identical.

While figure 1 (bottom) shows the naïve average fixation, figure 3a shows the same situation, where the original naïve fixation-to-word mapping path is plotted (in blue) and additionally re-computed fixation-to-word mappings are represented as dots on the vertical fixation time line `Ft`. Figure 3a shows the projection of fixations on the lines above and below the naïve default mapping. For fixation time `F1`, two additional fixation-to-word mappings are generated in the lines below the naïve mapping (in the progression graphs the words further down in the text appear higher in the graph), while for fixations times `F2` and `F3` are generated fixation-to-word mappings in the lines above and below the default one. Note that different fixation-to-word mappings at one fixation time may also be distributed in different windows. For instance, a `down` re-mapping of the fixation numbered 0 in figure 1 (top) at the bottom of the source window may be re-located in the top of the target window, while the `up` alternative would appear in the source window, e.g. on *decreased*, as shown in figure 2 (top), where the same fixation is numbered 1.

In a second step a path through the lattice of fixation-to-word re-mappings is re-computed based on the minimum penalty score of the distances between successive nodes. Assuming that a fixation-to-word mapping `n` is consolidated for a given fixation time `Ft`. A penalty score for each possible fixation-to-word mapping `m` at the next fixation time `Ft+1` and its fixation center `f` is computed by summing up a number of features as described below. The fixation-to-word mapping `m` with the lowest penalty score is then consolidated. Figure 3b shows the links to the three possible successor nodes, where the link in bold represents the strongest connection with lowest penalty score.

Different features are considered to compute the scores between successive mapping nodes, depending on whether the two successive fixation mappings occur in the same window, or whether the gaze moves from the source window to the target window or vice versa. In case two successive fixations occur in the same window (i.e. the source or target window), we assume that a sequence of text is being read so that the eyes move likely forward over the text. In case the eyes move from one window to the other, we assume that the eyes move (close) to the translation of the sequence that was previously looked at in the other window.

According to (Rayner, 1998), during “normal” reading the eyes jump in distances of around 5 to 15 characters along the text from left to the right, often skipping short function words. As drift of gaze data is the more unusual case, we assume that the measured gaze sample points which are received from the eye tracker, and thus the center of the various fixations that we compute from it, are close to the characters and words which are actually read. In addition, we assume that translators read a piece of text (in the source or the target window) which is close to currently translated sequence. These considerations are formalized in the following four functions:

- Cursor distance:

  C(n, m) = abs(CurPos(m)–CurPos(n) - 10)
- Source ID distance:

  S(n, m) = abs(STID(m) – STID(n) + 2) * K
- Last keystroke distance:

  L(m) = abs(STID(m) – STID(l)) * K
- Character-pixel distance:

  P(f, m) = EuclidDistance(f, m) / z
  i.e.: sqrt((f(x)–m(x))^2 + (f(y)–m(y))^2)/z

where n and m represent two fixation-to-word mappings, f is the fixation center of m, K=6 is approximately the average length in characters of (English) words, z=24 is the size of the characters on the screen that we used in these experiments, and l is the cursor position of the last character that was typed in the target window.

The SourceTextId STID(.) is computed based on the alignment between the source and target text. Words in the source and target target text are numbered, and the alignment information allows us to know the SourceTextId for each target word in the translation. This information can be spread out to the keystrokes which actually produce the target words and the target text. An algorithm described in (Carl, 2013) which describes how SourceTextIDs for keystrokes and fixations are computed from the alignments.

Between each consolidated fixation-to-word mapping n at Ft and every possible successor node m at the following fixation time Ft+1, a penalty score is computed as:

CSLP1(n,m,f) = C(n,m)+S(n,m)+L(m)+P(f,m)

Since sometimes the eye may slip up or down a line or two (particularly when switching between the two windows) we also compute the penalty score between the consolidated node o of the preceding fixation time Ft-1 and the successor mapping m, so as to skip the impact of the current, possibly slipped fixation on the gaze path:

CSLP2(o,m,f) = C(o,m)+S(o,m)+L(m)+P(f,m)

This situation is depicted in figure 3c. There is one consolidated node n at fixation time F1 which is connected to all three possible successor nodes m1..3 at fixation time F2. These connections are represented in fine dotted lines. In addition, there are also connections from the consolidated node o at fixation time F0 which link to the three possible fixation-to-word mappings in F2. These links are represented with dashed curved connectors. There are thus six penalty scores for three nodes m1..3 in F2. The node which the lowest penalty is consolidated, and the link to the previous consolidated fixation mapping is plotted in bold arrows in figure 3c. Even though the distance to its immediate predecessor node in F2 is quite large, the node in F2 was consolidated due to the similar SourceTextId which it shares with the consolidated fixation-mapping node in

F2 and its proximity to the previously typed character. The algorithm iterates through the expanded fixation-to-word mapping lattice. Once a fixation-to-word mapping is consolidated, the penalties of the next fixation nodes are computed as shown in figure 3c and so on until the end of the lattice is reached.

As mentioned previously, penalty scores are slightly differently computed if the two successive fixation-to-word mappings are in different windows. In this case we assume that the eyes seek to retrieve the translation of the word that was looked at (or worked on) in the other window, rather than proceeding in the text. That is, penalty score increases as the two successive fixations-to-word mappings return a different SourceTextID:

- Source ID window change:

  W(n, m) = abs(STID(m) – STID(n)) * K

Since shifting of windows is different from usual reading behaviour, we also do not assume that eyes move in jumps of around 10 characters, as above, and omit the cursor distance penalty function C(.). For window-changing sequences of fixations, we thus introduce two functions, analogous to the previous ones, where: WLP1(n, m, f) computes the penalty scores for immediate successive nodes and WLP2(o, m, f) computes the penalty scores for two nodes distance:

WLP1(n, m, f) = W(n, m) + L(m) + P(f, m)

WLP2(o, m, f) = W(o, m) + L(m) + P(f, m)

The brain usually prefers visual input from one eye which is referred to as the dominant eye. Accordingly, fixations computed with the gaze data of the dominant eye correspond more precisely to the visual input and hence reveal more accurately what the brain was actually processing. According to wikipedia (http://en.wikipedia.org/wiki/Ocular_dominance), approximately two-thirds of the population is right-eye dominant and one-third left-eye dominant. As explained above, we compute up, down, left, right and average fixation-to-word re-mappings. Since we do not know the eye dominance of our participants, the left and the right fixation-to-word mappings take into account the fact that the preferred visual input may be on the left or right side eye respectively. In addition, we frequently observe phenomena of gaze drift, where the observed gaze data is a line below or above the one that we think is plausible to assume the person was actually reading. The up and down fixation-to-word re-mappings take into account such gaze drifts by simulating a shifting of the observed fixation a line up or down. However, we can expect that the left, right, up or down mappings do not change from one fixation to the next: the dominant eye does not change from one fixation to the other fixation and gaze usually does not drift in short distances of time from the line above to the line below. We thus assume that these fixation mappings are stable over stretches of time. To take this constraint into account, the penalty score P(f,m) is hat up and down drifts

P(f,m) = 0, if (ReMap(n) eq ReMap(m)

or (ReMap(o) eq ReMap(m)

P(f,m) = EuclidDistance(f, m) / z, otherwise

where ReMap(x) returns one of the values up, down, left, right or average, according to the way how the fixation-to-word mapping was computed.

Evaluation

The assessment and evaluation of a fixation-to-word re-mapping method is problematic, since we cannot know for sure where translators really look on the screen when they translate. Hence an adjustment (manual or automatic) can always be wrong, and an objective function or a test set against which a re-mapping method could be evaluated may be troublesome to establish.

A number of cognitive models of the human translation process exist which give us an intuition of the observed translation process data and which may serve as a basis for an evaluation of re-computed fixation-to-word alignment patterns. For instance, Jakobsen (2011) has found indications of a recurrent “micro-cycle”, i.e. a processing pattern consisting of six steps, some of which can be skipped or repeated several times. The processing cycle starts with an act of comprehension, namely reading the chunk of ST which is about to be translated (step 1). The translator then shifts his/her gaze to the TT to locate the position where the TT is about to be produced (step 2). The translation is typed and monitored (steps 3 and 4), and the translator's gaze shifts back to the ST, where the relevant reading area is located and the current ST word is read again (steps 5 and 6) (Jakobsen 2011, 48).

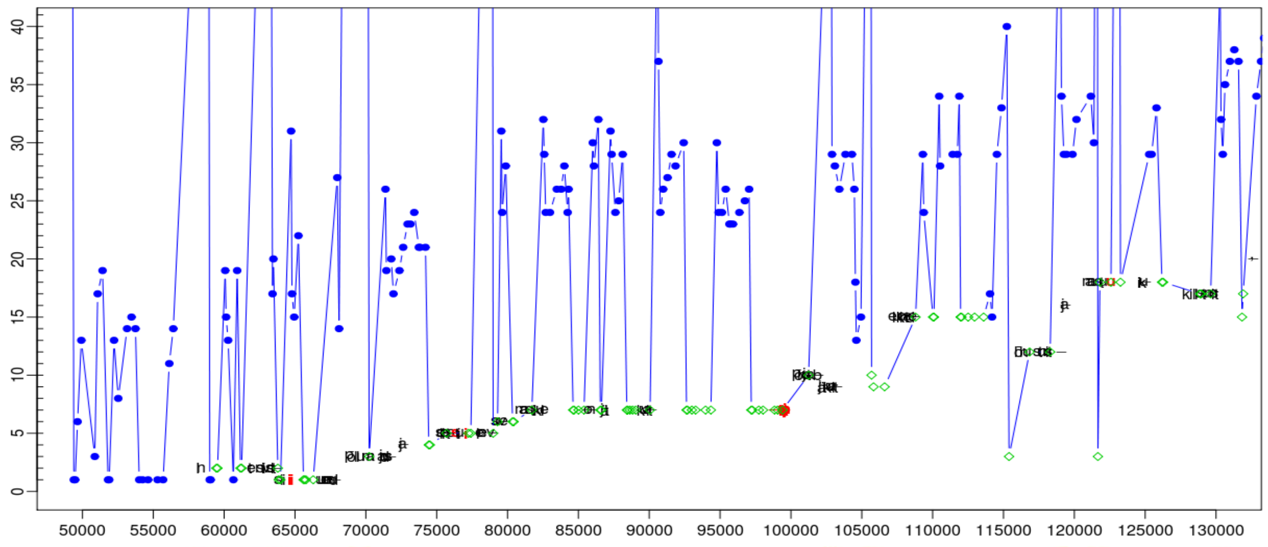
While such models give us a general picture of what we may expect in the translation activity data, they are still far from exactly predicting where the next fixation is to be expected. In addition, a large variation of individual translation styles has been described, for instance in (Dragsted & Carl 2013), so that the evaluation of the re-mapped log files remains, for the moment, subjective and intuitive. In the future we hope that in a real-time reactive application the success of a re-mapping method could be evaluated based on its usefulness, Figures 4 show an example of the present re-mapping algorithm.

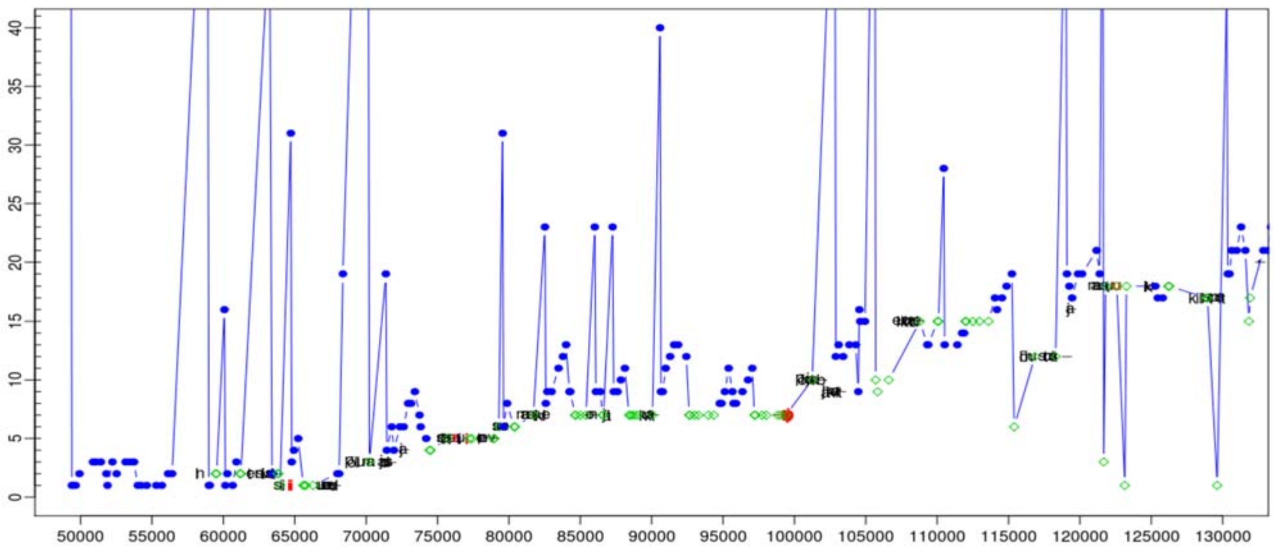
The progression graph in Figure 4a clearly shows a systematic drift of the source text fixation mappings about 12 to 20 words ahead of the translations on which the translator is working. Figure 4b shows a re-mapping which clearly comes closer to the initial main criteria which were previously established to design of the re-mapping algorithm:

– successive fixations are one or two words apart

– translators are likely to read source passages which they are currently translating

The re-mapped version also better accounts for the recurrent micro-cycle, as described above (Jakobsen 2011).





Figur 4a (top) naïve mapping vs. its re-mapped version 4b (bottom). Both figures represent the same translation segment.

References

Dragsted, B., & Carl, M. (2013). *Towards a classification of translation styles based on eye-tracking and keylogging data*. Journal of Writing Research, in press.

Hornof, A. J., & Halverson, T. (2002). *Cleaning up systematic error in eye-tracking data by using required fixation locations*. Behavior Research Methods, Instruments, & Computers, 34, 592–604.

Jakobsen, A.L. (2011). T*racking translators' keystrokes and eye movements with Translog*. In C. Alvstad, A. Hild & E. Tiselius (Eds.), Methods and Strategies of Process Research (pp. 37-55). Amsterdam: John Benjamins.

Abhijit Mishra, Michael Carl, Pushpak Bhattacharya (2012). *A heuristic-based approach for systematic error correction of gaze data for reading,* Workshop on Eye-tracking and Natural Language Processing, Coling 2012, 24th International Conference on Computational Linguistics, 15 December, 2012 Mumbai, India

Rayner K. (1998) *Eye movements in reading and information processing*: 20 years of research. Psychol Bull. 1998 Nov; 124(3):372-422.

Špakov, O. (2007). GWM – the Gaze-to-Word Mapping Tool, available online at http://www.cs.uta.fi/~oleg/gwm.html.

Feature Representation in the Translation Process Research DB

Abstract

For more than 10 years CRITT has been involved in Translation Process Research (TPR). TPR-data was collected by the Translog tool and released in 2012 as a Translation Process Research Database (TPR-DB). Within the CASMACAT project, data for post-editing machine translation is collected and added to the TPR-DB. The second release of the TPR-DB contains more than 900 translation sessions accumulating more than 300 hours of recorded translation sessions. This paper describes the features and visualization options of the recent release of the Translation Process Research DB. This database contains recorded logging data, as well as derived and annotated information. Seven kinds of simple and compound process- and product units are described which are suited to investigate human and computer-assisted translation processes and for advanced user modeling.

1 Introduction

Since 2006 CRITT has developed a data acquisition software, Translog (Jakobsen and Schou, 1999, Carl 2012) with which translators' keystroke and gaze activities can be recorded. This tool is now the most widely used tool of its kind (Jakobsen, 2006). In contrast to previous think - aloud elicitation methods, a keylogger runs in the background so as not interfere with the writing or translation process. Translog-II (Carl, 2012a), which is its most recent implementation, logs the exact time at which each keystroke operation is made and in a reply mode the translation session can be played back. If connected to an eye-tracker, Translog-II also records gaze-sample points, computes gaze fixations and maps the fixations to the closest character on the screen. The information is stored in an XML format and can be analyzed in external tool (Carl, 2012).

While Translog was originally designed to investigate reading, writing and translation processes, it was recently also extended to record post-editing sessions of machine-translation output. However, Translog-II does not provide an ecologically valid working environment, with which post-editors are used to work in their daily environment. Translog-II presents two running text in a source and target window, while modern translation aides, such as translation memories, segment the texts into fragments and present a source segment with its translation in a more structured manner. In order to obtain a more realistic picture of professional translators' working styles and to assess how to support their translation processes with advanced machine translation technology, the CASMACAT project seeks to implement an advanced state-of-the art browser-based post-editing environment and combine this with Translog-II style keyboard logging and eyetracking possibilities. In this way, detailed empirical data can be collected from a realistic translation environment, and the assessment of this data may lead to a more complete picture of human computer-aided translation processes.

The structure and content of the collected data from Translog-II and from CASMACAT is different in terms of the attributes and containers, but the logged data undergoes a similar compilation process (Carl, 2012b) to generate a number of tables, which can then be used as a basis for further analysis (Carl et al. Forthcoming, Balling and Carl 2013, Elming et al, forthcoming; Wilker et al, forthcoming) and visualization (e.g. Carl, Dragsted & Jakobsen, 2011). The logged data together with the derived tables is released in the form of Translation Process Database (TPR-DB), which can be freely downloaded from http://bridge.cbs.dk/platform/?q=node/18

As in the first version of this data (Carl, 2012b), also the current version of the CRITT TPR-DBv1.3 contains for each translation session seven different types of units. The logging data provides two basic types of data: text modifying keystroke data (KD), that is insertions and deletions, and fixations on the source or target text (FD). From these basic units are derived two more complex processing units: text production units (PU) and fixation units (FU) which represent respectively sequences of coherent writing and reading. In addition to these production units, the TPR-DB contains three text-based units, which are derived from the final translation product: source text tokens (ST), target text tokens (TT) and alignment units (AU):

1. Keystrokes: basic text modification operations (insertions or deletions), together with time of stroke, and the word in the final text to which the keystroke contributes.
2. Fixations: basic gaze data of text fixations on the source or target text, defined by the starting time, end time and duration of fixation, as well as character offset and word index of fixated symbol in the source or target window.
3. Production units: coherent sequence of typing (cf. Carl and Kay, 2011), defined by starting time, end time and

duration, percentage of parallel reading activity during unit production, duration of production pause before typing onset, as well as number of insertion, deletions.

4. Fixation units: coherent sequences of reading activity, including two or more subsequent fixations, characterized by starting time, end time and duration, as well as scan path indexes to the fixated words.
5. Source tokens: as produced by a tokenizer, together with TT correspondence, number, and time of keystrokes (insertions and deletions) to produce the translation, micro unit information.
6. Target tokens: as produced by a tokenizer, together with ST correspondence, number, and time of keystrokes (insertions and deletions) to produce the token, micro unit information, amount of parallel reading activity during.
7. Alignment units: transitive closure of ST-TT token correspondences, together with the number of keystrokes (insertions and deletions) needed to produce the translation, micro unit information, amount of parallel reading activity during AU production, etc.

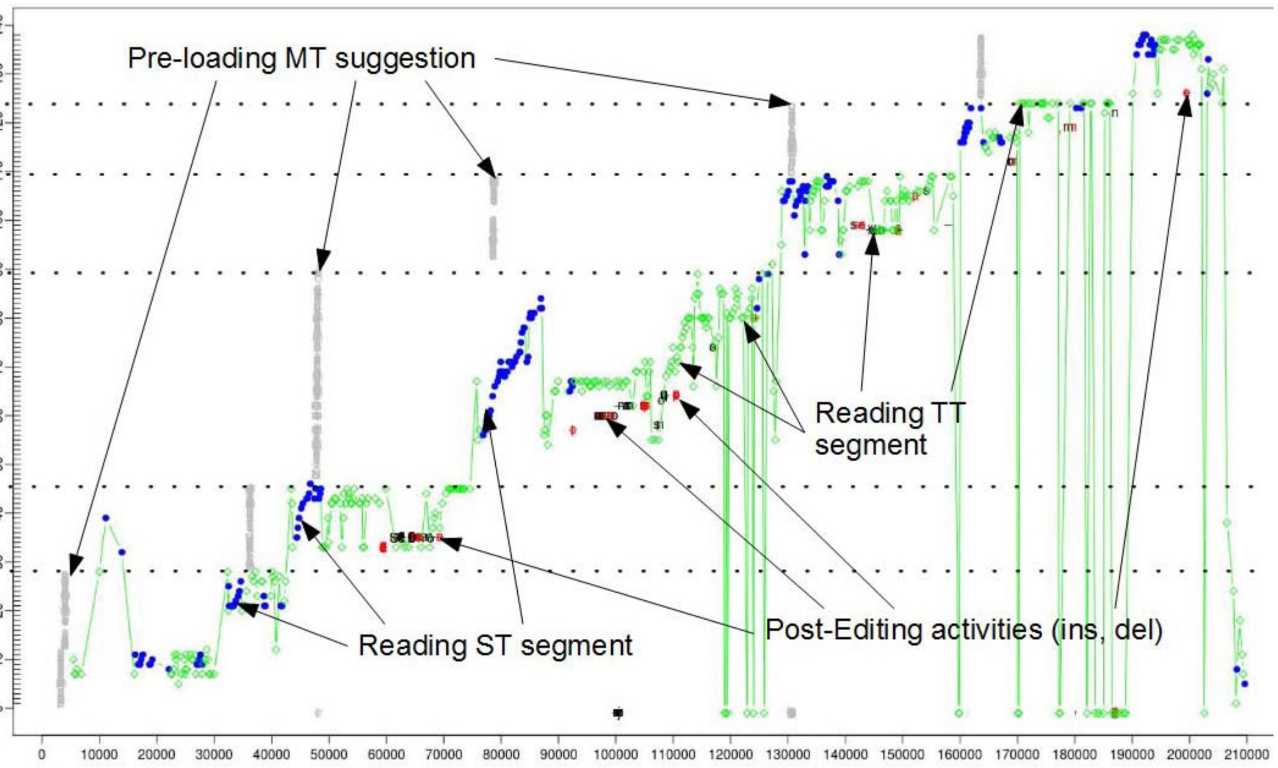


Figure XX shows a visualization of the keystroke and fixation data, as collected in a post-editing session of the Casmacat Prototype-2 which involves a text of approximately 140 words of six segments. Translation progression graphs visualize how translations emerge in time, enumerating the source text words on the vertical axis and the translation time on the horizontal axis. We have adapted the visualization of translation progression graphs, which was developed for Translog-II logging data to both, the Casmacat Prototype-1 and the Casmacat Prototype-2.In addition to the original Translog-II logging data, the CASMACAT workbench plots the source and the target text in the form of segments, and produces automatic translations, as well as deletions and insertions in the interactivity mode. Accordingly, the representation and visualization schema was extended to take these additional features into account. The vertical axis enumerates the source text words (0.. 140) and the horizontal axis shows the time in which the translations of the source text were produced. The dotted lines divide the six segment, which are sequentially (pre) loaded, filled into the target buffer and then edited. The various symbols in the graph represent:

- blue diamonds represent fixations on the source text
- green diamonds represent fixations on the target text
- black characters represent insertions
- grey characters represent automatic insertions
- red characters represent deletions

The graph shows the temporal sequence of when segments are loaded into the target buffer, when and where translators read the source segments and the translations, and when which MT suggestions are modified. The unit information in the TPR-DB tables is thus instrumental to analyze and visualize translation processes. Much more information is contained in the TPR-DB tables, besides the one plotted in figure XX. This paper describes the features

The paper describes the units and the features[1] that are extracted from from logged and annotated data. Section 2 describes the two basic keystroke and fixation units. Section 3 illustrates examples of the derived production and fixation units. A special property of those units is parallel and alternating reading and typing behavior which indicates workload of the translator. The idea and the way to assess this property is described in section 4. Section 5 looks into characteristics of units that can be automatically derived from the final translation product: source tokens, target tokens and alignment units. Section 6 exemplifies how the translation construction of these production units can be decomposed into several micro units.

2 Basic units

For each translation session, the TPR-DB contains seven tables, each of which identifying a different type of unit. The first column in each TPR-DB table is an identifier of the event or unit (KEYid, FIXid, FUid, STid, TTid, PUid, AUid). Successive columns encode various features which characterize the event or unit.

KEYid	Time	Type	Cursor	Char	STid	TTid
0	92016	ins	0	E	2	1
1	92172	ins	1	l	2	1
2	92313	ins	2	_	2	1
3	92375	ins	3	e	2	2
4	92563	ins	4	n	2	2
5	92828	ins	5	f	2	2
6	92938	ins	6	e	2	2
7	93047	ins	7	r	2	2
8	93266	ins	8	e	2	2
9	93610	del	8	e	2	2
10	93797	ins	8	m	2	2
11	93875	ins	9	e	2	2
12	93938	ins	10	r	2	2
13	94078	ins	11	o	2	2

Table 1: Keystroke information

2.1 Keystroke data

The Keystroke tables encode single events in time with no duration. All other TPD-DB tables encode textual or temporal units which stretch over parts of one or more words and which have at least one starting time and a duration, as described below.
As shown in Table 1, keystrokes have a Time at which they were produced, a Type, indicating whether it was an insertion or deletion, a position in the text (a Cursor offset) at which the text was modified, the actual character (Char) which was inserted or deleted, as well as the target text token (TTid) to which the keystroke has contributed and the source text token (STid) of which the TTid is the translation. Note that the TTid refers to the token in the final text.
During a fixation, the gaze is maintained on a single location. Reading involves fixating on a successive locations across a text, but neither is the eye perfectly steady during fixations, nor do the eyes move smoothly over a text. There are many methods to compute fixations. In Translog-II we currently use a density-driven fixation computation algorithm, which clusters gaze samples within a distance of 60 pixels into a single fixation, if the duration is longer than 40ms. The center of the fixation is then mapped on the closest character using build-in functions.

FIXid	Time	Win	Dur	Cursor	STid	TTid
251	93921	2	250	7	2	2
252	94171	2	150	9	2	2
253	94374	1	183	65	10	13
254	94546	1	267	25	4	5
255	94937	1	100	26	4	5
256	95077	1	184	25	4	5
257	95671	2	400	15	1	3
258	96062	1	316	791	152	170
259	96374	2	200	13	1	3
260	98765	1	217	25	4	5
261	98984	1	283	36	6	6
262	99265	1	217	24	4	5
263	99499	1	100	17	3	4
264	99624	1	116	17	3	4
265	99812	1	982	26	4	5
266	101562	1	1199	32	6	6
267	103812	2	299	32	4	5
268	105780	1	200	38	6	6
269	105999	1	117	425	82	86
270	108062	1	133	185	33	42
271	108359	1	100	54	8	8+9
272	108452	1	333	179	31	39
273	108796	1	133	295	54	62
274	109077	1	200	51	8	8+9
275	109452	1	117	58	9	12

Table 2: Fixation information

2.1 Fixation data

The table in Table 2 indicates the beginning of a fixation (Time) and its duration (Dur). The fixation table shows in which window (Win) a fixation was detected, 1 for source text window and 2 for the target text window and the Cursor offset of the closest character at which the center of the fixation was detected. While the cursor offset refers to the text as it emerges, the STid and TTid refer to the source and target text tokens of the final text. Thus at a certain time during text production cursor position 5 of the TT may for instance contain an "a" which is part the word "asesino". The fixation will be assigned TT4 if "asesino" turns out to be the 4th word in the final translation, irrespectively of where in the text this word occurred when it was fixated. In this way we can count the number of fixations on one word, even if the word changes its locations in the text during the editing process. Note, however, that the precision of this information has to be handled with care, since 1. movements of text fragments, particularly deletions, can be traced only very imprecisely, and 2. fixations and their mapping on the symbols may be quite noisy, due to different reasons of fixation drift.

1 Some of the features are only available in the CRITT TPR-DB V1..1

3 Process Units

3.1 Production units

Production units (PUs) are sequences of coherent typing activity (cf. Carl and Kay, 2011). A production unit boundary is defined as a delay of 1000ms or more without keyboard activity. It is assumed that coherent typing is interrupted beyond this delay of time, with a likely shift of attention towards another text segment. As a coherent temporal/textual segment PUs have a temporal beginning (Time) and a duration (Dur), and as they cover one or more insertion or deletion keystrokes (Edit operations) which contribute to build up one or more target text tokens (TTid). In the example in Table 3, the sequence:

El_enfere[e]mero_asesiono_re[er_ono]no_recibe

PUid	Time	Dur	Pause	ParalS	ParalT	Ins	Del	STid	TTid	Edit
0	92016	7250	1140	900	0.00	34	7	1+2+3	1+2+3+4	El_enfere[e]mero_asesiono_ re[er_ono]no_recibe
1	100406	1313	1875	562	0.00	8	0	3+4	4+5	_cuatro_
2	103594	4187	13735	299	0.00	23	3	4+5	5+7	sentencias_de_vida.__[__.]__

Table 3: Production units

was typed within 7250ms, starting at time 92016 with no inter-key delay of more than 1000ms. A delay (Pause) of 1140ms follows this typing sequence before the next PU starts at Time 100406ms. The table 3 also indicates the number of insertions and deletions of the PUs. PU_0 contains 34 insertions and 7 deletions. The latter are within square brackets and must be read in the reverse direction. Thus, the substring "[er_ono]" is actually the deletion "ono_re" which reflects the correction of:

asesiono_re --> asesino_recibe

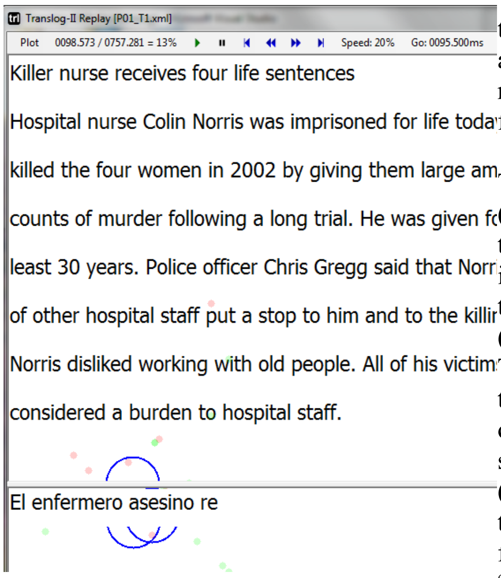
Note that PU_1 "_cuatro_" accounts for two target words (TT_{4+5}), as the blank, represented by an underscore "_" already counts as part of the next word. Table 3 also indicates where and how long the translator looked at the screen while typing the translation. The feature ParalS and ParalT give the amount of time the translator was looking at the source and the target window respectively while producing the translation. That is, during the 7250ms that it took to produce PU_0, the translator looked almost 1sec (900ms) at the source text window, but did not look at the target window.

3.2 Fixation Units

FUid	Time	Dur	Pause	ParalK	Path
11	93921	1340	410	1340	2:2+2:2+1:10+1:4+1:4+1:4+
12	95671	903	2191	903	2:3+1:152+2:3+
13	98765	2029	768	888	1:4+1:6+1:4+1:3+1:3+1:4+
14	108062	1507	665	0	1:33+1:8+1:31+1:54+1:8+1:9+

Table 4: Four fixation units

Similar to PUs, Fixation Units (FUs) indicate sequences of coherent reading behavior. Based on experimental evidence (Carl and Kay, 2011) we define a boundary between two successive FUs if a gazing pause is longer than 400ms. That is, if the stream of gaze samples indicates the gaze directs away from the screen for more than 400ms, thus interrupting coherent reading activity, we assume a boundary of a fixation unit and the beginning of the next fixation. This may happen, for instance, when the gaze is shifts away from the screen to the keyboard, or to some other places.


Translog-II Replay [P01_T1.xml]

Plot 0098.573 / 0757.281 = 13% Speed: 20% Go: 0095.500ms

Killer nurse receives four life sentences

Hospital nurse Colin Norris was imprisoned for life toda

killed the four women in 2002 by giving them large am

counts of murder following a long trial. He was given fo

least 30 years. Police officer Chris Gregg said that Norr

of other hospital staff put a stop to him and to the killir

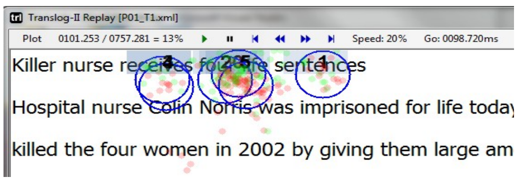
Norris disliked working with old people. All of his victim

considered a burden to hospital staff.

El enfermero asesino re

Figure 1: Screen shot of replay situation FU_{12}

Table 4 shows four FUs (FU_{11} to FU_{14}). As with the PUs, the Time indicates the beginning of the FU while the duration (Dur) indicates its length.


Translog-II Replay [P01_T1.xml]

Plot 0101.253 / 0757.281 = 13% Speed: 20% Go: 0098.720ms

Killer nurse receives four life sentences

Hospital nurse Colin Norris was imprisoned for life today

killed the four women in 2002 by giving them large am

Figure 2: Screen shot of replay situation FU_{13}

The fixation path is a sequence of fixations on the source window (1) or the target window (2) and the word ID looked at. The path consists of one or more fixations indicated by a tuple "Window:WordID" where successive fixations are separated by a "+". The first FU in Table 4 (FU_{11}) shows a sequence of six fixations, first on the second word in the target window "enfermo" (2:2), followed by a number of fixations on fourth source word "four" (1:4). On the way from the target text word "enfermo" to the source text word "four", a fixation on word 10 "Colin"

was recorded, which is just one line below the "four". Figure 1 shows the a screen shot of the Translog-II replay at time 98573, just before the start of the third FU. FU12 comprises of three fixations (marked by a blue circle), two of which are on word 3 "asesino" in the target text, while one fixation is at the end of the source text on word 152. While this accounts for the measured gaze data, it is more likely that a slight drift causes the second fixation is mapped into the ST window, while the translator was actually looking at the ST word.
The third fixation unit in Table 4, FU13 is plotted in Figure 2 and represents a reading sequence of the title (Killer nurse receives four life sentences). It shows how the eyes go back and forth between word 6 ("sentences"), 4 ("four") and 3 ("receives"). As it is not particularly difficult to understand the meaning of the sequence of words, the long reading time of more than 2 seconds (2029ms) suggests that a process of pre-translation takes place during ST reading, in which the translator reflects on how the translation should be rendered.
Note that the sum of all FU durations may be longer than the sum of all fixation durations, since FUs include inter fixation delays shorter than 400ms which may not be part of any fixation.

3.3 Parallel and alternating reading and writing

Similar to the ParalS and ParalT features in the PU tables, the ParalK feature in the FU table indicates the amount of parallel keyboard activity. This FU11 and FU12 take place while the translator is at the same time writing, while no keyboard activity was obserbed during FU14.
Figure 3 illustrates the overlap of reading and writing activity. It puts into relation the source text (vertical axis) and the translation time (horizontal axis). Insertions are represented in black letters, deletions are red. The progression graph in Figure 3 plots the keystroke data of Table 1, the fixation data from Table 2, as well as the three production units of Table 4 and four fixation units from Table 3. The first part in Figure 3 (approx. Time 92000ms to 94000ms) reproduces the production of words 1 and 2 ("El enfermero") as plotted in Table 1. The linked blue x-es represent the fixations (Table 2). The red horizontally striped boxes indicate PUs while the green boxes represent FUs.

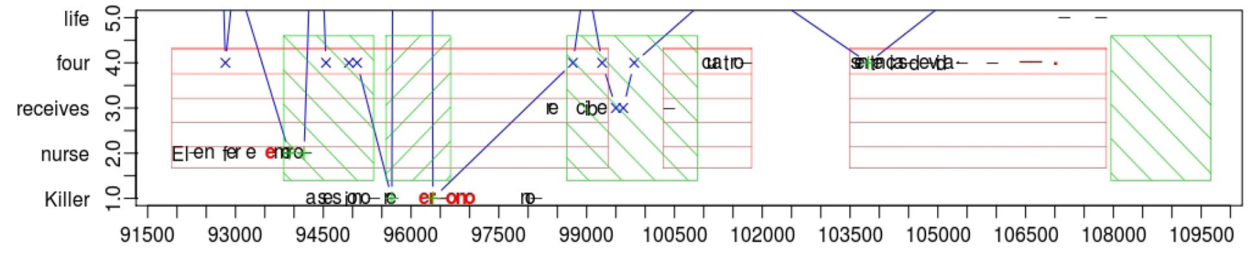


Figure 3: The progression graph shows information from Tables 1 to 4

Reading and writing activity can go on concurrently in parallel. For instance, the FU11 between Time 93921-95260 and FU12 between 95671 -- 96574 take place while the translator performs a coherent typing activity at the same time generating PU0. While FU11 and FU12 overlap 100% with PU0, FU13 between Time 93921-95260 only partially overlaps with two adjacent PU0 and PU1. While there is 43.81% overlap with production activity of FU13, FU14 has no overlap at all. Progression graphs, as in Figure 1 may thus illustrate in a graphical manner the relation between reading and writing activities.

4 Product Units

Besides fixation and production units, there are three more units in the TPD-DB tables: Source Token (ST), Target Token (TT) and Alignment Units (AU).

4.1 Alignment Units, Source and Target Tokens

AUid	AUtarget	AUsource	SL	TL	Study	Person	Text	Task
44	de	of	en	es	BML12	P01	1	T
45	tranquilizantes	sleeping_medicine	en	es	BML12	P01	1	T

Table 5: Alignment unit

Source and target tokens correspond to sequences of characters, usually separated by a blank, while AUs refer to m-to-n source-to-target token correspondences. The tables provide similar kind of information for these three different kinds of units. These tables contain various information concerning the source/target correspondances, who and how the translation was produced, and information concerning the session.

AUid	Session	Draft	Revise
44	757281	92016	290391
45	757281	92016	290391

Table 6: Session information

Table 5 shows three English --> Spanish AUs: the column AUtarget contains the TL string, while AUsource has the corresponding SL string. The column "Study" gives the name of the study, "Person" indicates the study unique identification of the translator, the "Text" column indicates which text was translated, and "Task" gives the kind of text production (T: translation, P: post-editing, E: editing).
Table 6, 7 and 8 are continuations of the AU information. Table 6 gives session information, Table 7 (macro unit) production information and Table 8 decomposes the macro unit in Table 7 into various micro units.
In Table 6, the column "Session" indicates the duration to the translation/post-editing/editing session, "Draft" shows the

lapse of time before the first keystroke was typed, i.e. the end of the orientation phase and beginning of the drafting phase, while "Revise" indicates the time when the drafting phase ended and the revision phase started. This is defined as the end of the first micro unit in which the last token of the text was translated (cf Jakobsen, 2002).

4.2 Typing Inefficiency

While Table 5 indicates for AU44 and AU45 that the final translation was "de" and "tranquilizantes" respectively, table 7 shows in the "Edit" column that first "de medicinas para dormir" was typed and later "medicinas para dormir" was again deleted. The table shows the overall number of keystrokes produced: there were 24 insertions, of which 21 characters (the string in square brackets) were later deleted. Even though "medicinas para dormir" and "tranquilizantes" are paraphrases, the former is part of AU44, since deletions are attributed to the preceding word. The time needed to type the translation is given by the duration feature (Dur).

AUid	Ins	Del	Dur	Cross	GazeT	GazeS	InEff	Edit
44	24	21	11407	1	549	200	15	de_medicinas_para_dormir[rimrod_arap_sanicidem]
45	15	0	1610	2	566	1963	0.94	tranquilizantes

Table 7: AU production information

The editing inefficiency measure (InEff) is the ratio of the number of produced characters divided by the length of the final translation. This is equivalent to the number of insertions and deletions divided by their difference:

InEff = Insertions + Deletions / Insertions – Deletions +1, where Insertions ≥ Deletions ≥ 0.

In most of the cases, the length of the final string in the translation product is equal the number of insertions – deletions + 1. We add 1 since the white space following the word is counted as being part of it. However, in some cases, no white space follows a words, in which case the InEff value can be smaller than 1.Thus, for AU44 in table 7 the number of the insertion and deletion keystrokes amounts to 45 which, divided by the length 3 of the final word "of " (including the following white space charater), results in an editing inefficiency of 15, while the number of keystroke string to produce "tranquilizantes" in AU45 amounts to the length of the final translation, and thus the editing effort is 0.94. Note that for post-editing the InEff can be 0 if a MT proposal was accepted without any modifications, while it would be 2 it the word was deleted and another word of identical length was retyped.

GazeT and GazeS indicate the total amount of gaze time on the source unit and the target unit respectively. In contrast to the "Paral" feature in Tables 3 and 8 this is not necessarily during translation production.

4.3 Micro units

Source and Target tokens, as well as AUs may be characterized by the number and type of micro units by which the translations are constructed. Alves and Vale (2012) refers to recurring editing activities of the same word translations as micro units. For them, "a micro TU is defined as the flow of continuous TT production ... separated by pauses during the translation process". A macro unit, then is a collection of micro units "that comprises all the interim text productions that correspond to the translator's focus on the same ST segment". The TPR-DB computes a micro unit as a coherent typing activity which contributes to the translation of the source or target token, or a AU. While there can be, in principle, any number of micro units (a translator can revise a piece of text very often), only information of the first two micro units is explicitly listed. Tables 8 shows the micro unit information for AU44 and AU45, while their macro unit information is given in table 7. The micro unit is characterised by the actual typing activity (Edit), the starting Time and duration (Dur) of the typing activity, the pause preceding that typing activity, and the amount of parallel reading and writing activity (Paral).

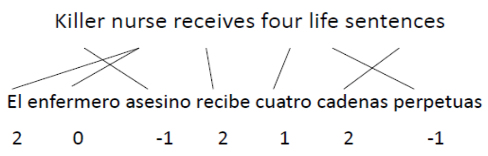
AUid	Edit1	Time1	Dur1	Pause1	ParalS1	ParalT1
44	de_medicinas_para_dormir	225703	11110	187	965	149
45	tranquilizantes	570250	1610	172	0	669
	Edit2	Time2	Dur2	Pause2	ParalS2	ParalT2
44	[rimrod_arap_sanicidem]	569781	297	22937	0	281
45	---	0	0	0	0	0

Table 8: Micro unit1 and micro unit2

Table 8 decomposes the production activity in Table 7 into two micro units: at Time 225703 the translator first types "de medicinas para dormir" in AU44. During a revision more than 4 minutes later, at time 569781 in micro unit2, the string "medicinas para dormir" is deleted and replaced by "tranquilizantes" at Time 570250 which is part of AU45, micro unit1. The duration of those activities is indicated, together with the pause following it and the parallel activity as described in section 4. Given the information in Table 6, we know that revision phase started in this translation session at time 290391, we see that micro unit 1 in AU44 takes place during translation drafting, while micro unit2 of AU44 and AU45 micro unit 1 are both revision events.

4.4 Cross value

The Cross feature represents alignment information in a procedural manner. It indicates how many words need to be consumed in the source text to produce the next word in the translation output. The assumption is that the source text is processed word by word from left to right (or from right to left) thereby emitting target words in the order they appear in the target text, following the ST-TT alignment links. The minimum number of words moved in the ST to produce the TT represents the cross value. Figure 9 gives an example from an English → Spanish translation: in order to produce the first Spanish TT word (El), two English words (Killer nurse) have to be consumed, which results in a cross value of 2 for "El". The second source word (nurse) emits two adjacent TT words. No further ST word has, thus, to be consumed to produce "enfermo", which results in a cross value is 0. To produce the third Spanish word, "assesino", one ST word to the left of "nurse" has to be processed together with a cross value of -1. Spanish "recibe" is the translation of two ST words to the right, "cuatro" one ST word ahead etc. and the respective cross values of 2 and 1 are emitted. The more syntactic reordering between source and target text take place the higher the average cross value will be. In case of a monotoneous translation, all cross values are 1.


Killer nurse receives four life sentences
El enfermero asesino recibe cuatro cadenas perpetuas
2 0 -1 2 1 2 -1

5 Conclusion

The paper describes several units and their feature characteristics in the CRITT TPR-DB. We hope that this can be a solid basis for future translation process research.

References

Alves, Fabio, Daniel Couto Vale, 2012, *On drafting and revision in translation: a corpus linguistics oriented analysis of translation process data.* Translation: Corpora, Computation, Cognition. Special Issue on the Crossroads between Contrastive Linguistics, Translation Studies and Machine Translation. Volume 2, Number 1. July 2012.www.t-c3.org

Balling, Laura Winther and Michael Carl. Forthcoming. Production time across languages and tasks: a large-scale analysis using the critt translation process database

Carl, Michael (2012a). Translog-II: a Program for Recording User Activity Data for Empirical Reading and Writing Research, Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC12),

Carl, Michael (2012b). "The CRITT TPR-DB 1.0: A Database for Empirical Human Translation Process Research". Proceedings of the AMTA 2012 Workshop on Post-Editing Technology and Practice (WPTP 2012). ed. / Sharon O'Brien; Michel Simard; Lucia Specia. Stroudsburg, PA : Association for Machine Translation in the Americas (AMTA), 2012. p. 9-18.

Carl, Michael, Silke Gutermuth and Silvia Hansen-Schirra. Forthcoming. "Post-editing machine translation – a usability test for professional translation settings". In *Psycholinguistic and cognitive inquiries in translation and interpretation studies*, edited by John W. Schwieter and Aline Ferreira. Cambridge Scholars Publishing

Carl, Michael and Kay, Martin, 2011, *Gazing and Typing Activities during Translation : A Comparative Study of Translation Units of Professional and Student Translators*. I: Meta, Vol. 56, Nr. 4, 2011, s. 952-975.

Carl, Michael; Barbara Dragsted; Arnt Lykke Jakobsen / A Taxonomy of Human Translation Styles In: Translation Journal, Vol. 16, No. 2, Tralogy 2011. Translation Careers and Technologies: Convergence Points for the Future, Paris, Frankrig. 2011

Elming, Jakob, Michael Carl, and Laura Winther Balling. Forthcoming. "Investigating User Behaviour in Post-editing and Translation Using the CASMACAT Workbench." In *Expertise in Post-editing: Processes, Technology and Applications*, edited by Sharon O'Brien, Michael Simard, Lucia Specia, Michael Carl and Laura Winther Balling. Cambridge Scholars Publishing.

Jakobsen, Arnt Lykke (2002) "Translation drafting by professional translators and by translation students." Empirical Translation Studies: Process and Product. Copenhagen Studies in Language 27, 191-204.

Jakobsen, A. L. 1999. Logging target text production with Translog. In Hansen, G. (ed.), *Probing the process in translation: methods and results*, *Copenhagen Studies in Language*, volume 24. Copenhagen: Samfundslitteratur. Pages 9–20.

Wilker Aziz, Maarit Koponen and Lucia Specia . Forthcoming. "Post-editing time and cognitive effort." In *Expertise in Post-editing: Processes, Technology and Applications*, edited by Sharon O'Brien, Michael Simard, Lucia Specia, Michael Carl and Laura Winther Balling. Cambridge Scholars Publishing.