The ACCEPT Post-Editing Environment: a Flexible and Customisable Online Tool to Perform and Analyse Machine Translation Post-Editing

Johann Roturier, Linda Mitchell, David Silva
Symantec Ltd.
Ballycoolin Business Park
Blanchardstown, Dublin 15
Ireland
{johann_roturier,linda_mitchell,david_silva}@symantec.com

Abstract

This paper presents an online environment aimed at community post-editing, which can record and store any post-editing action performed on machine translated content. This environment can then be used to generate reports using the standard XLIFF format with a view to provide stakeholders such as machine translation providers, content developers and online community managers with detailed information on post-editing actions. This paper presents in detail the functionality available within the environment as well as the design choices that were made when creating this environment. Preliminary usability feedback received to date suggests that the feature set is sufficient to perform community post-editing.

1 Introduction

Machine translation (MT) systems are increasingly used to produce rough translated versions of documents that may be reviewed and possibly modified by post-editors in order to produce improved versions. For instance, Dell and Welocalize recently announced an MT-based localisation program for the translation of Web content using 27 MT engines.[1] Thanks to this deployment, machine-translated documents (such as technical support articles) are made available to end-users who may glean useful information if their knowledge of the source language is not sufficient and if the quality of the MT output is sufficient. In some cases, however, previous studies have shown that the quality of MT output may not be sufficient for the output to be found comprehensible by end-users (Roturier and Bensadoun, 2011), especially when the source text is uncontrolled (Mitchell and Roturier, 2012). This is especially true when source content is user-generated, which is why the ACCEPT project aims at developing new technologies designed specifically to help MT work better in an online community environment.[2]

Publishing documents that are difficult or impossible to understand defeats the purpose of publishing documents in the first place, so post-editing these documents before (or just after) they have been published is often considered as an important step in document localisation production workflows (Flournoy and Duran, 2009). Providing post-editors with the right environment to perform this activity has received a lot of attention lately since post-editing is a very different task from translation. This paper, which presents an online environment aimed at community post-editing, is divided as follows: Section 2 reviews existing post-editing environments by highlighting missing functionality from a community perspective. Section 3 presents the choices that were made when designing the ACCEPT environment and Section 4 describes its various functionality. Section 5 presents the results of a small evaluation study conducted with the help of an online survey, seeking to elicit feedback from users of the environment. Finally conclusions and suggestions for future work are presented in Section 6.

[1] http://www.welocalize.com/dell-welocalize-the-biggest-machine-translation-program-ever/

[2] http://www.accept-project.eu/

Sharon O'Brien, Michel Simard and Lucia Specia (eds.)
Proceedings of MT Summit XIV Workshop on Post-editing Technology and Practice, Nice, September 2, 2013, p. 119–128.

2 Related work

MT technology is increasingly used by Language Service Providers (LSP), as revealed by a 2009 TAUS market survey, which showed that 40% of the surveyed LSPs already used MT, with a majority of the remaining 60% indicating that they were considering an MT integration in their processes in the next two years.[3] The reasons for the increase in MT technology adoption are varied. One obvious reason concerns the productivity gains in the translation industry reported by several studies, such as Plitt and Masselot (2010), de Almeida and O'Brien (2010), Guerberof Arenas (2012) and (Green et al., 2013). Another reason is related to the increasing ubiquity of machine translation tools (such as Google Translate) and the proliferation of tools providing an environment where machine-translated text can be improved. Such environments include desktop-based standalone tools, web-based generic environments and web-based dedicated environments.

Desktop-based tools include generic computer-aided tools (CAT) or translation environment tools (TenT) that have been enhanced to support the post-editing of machine-translated content. Such tools tend to be aimed at professional translators, rather than community users, so they will not be reviewed any further. Dedicated standalone post-editing tools aim at studying the work of post-editors, for example by recording post-editing actions thanks to keylogging or eye-tracking software. Examples of such tools include Translog II (Carl, 2012), PET (Aziz et al., 2012) or iOmegaT [4]. Our work differs from these standalone tools since the reports generated by our environment are more concise and provide a summarising overview of keylogging actions per revision, per segment, per task for each participant.

Web-based dedicated tools are tightly integrated with the platform where the source and target content is created and published. An example of such an environment is the wikiBABEL platform (Kumaran et al., 2008), which provides a user interface and linguistic tools for collaborative correction of the rough content by a community of users, thus helping the creation of improved content in the target language. While the ACCEPT environment also targets user communities, it is not dedicated to a single community of users (such as Wikipedia). As described in section 3, our environment is currently available by logging to a portal, but its architecture is sufficiently flexible that it could in theory be used in any content management environment.

Web-based generic tools allow users to log on to an online system to post-edit machine-translated content. Examples of such environments include MateCat [5] and TransCenter (Denkowski and Lavie, 2012). These tools allow translators to log on to a web-based translation editor to view sentences in a simple, easy-to-follow grid format. Both tools allow project data to be exported in HTML or comma separated value (CSV) format. Our work differs from these environments with respect to two main points: the ACCEPT environment is aimed at community users so a grid format to display source and target texts does not seem appropriate. As described in Section 4, our user interface focuses on the target text. Our environment also supports the export of data in XLIFF format instead of CSV format to maximise interoperability. Another tool that falls in this area is CASMACAT, which is a sophisticated tool that aims at investigating the integration of technology in translation using logging and eye-tracking technology (Elming and Bonk, 2012). The ACCEPT tool differs from CASMACAT in many aspects, including the technology used for the client application. While CASMACAT uses both HTML5 and JavaScript in its client application, the ACCEPT client application is written in JavaScript and uses JQuery libraries.

Finally it is worth mentioning the Microsoft Collaborative Translation Framework, [6] which is a hybrid web-based method enabling specific users to submit corrections and to retrieve translation candidates.[7] While this framework provides users with the ability to retrieve contributed segments in real time, it does not allow for any other informa-

[3] https://www.taus.net/reports/lsps-in-the-mt-loop-current-practices-future-requirements

[4] http://try-and-see-mt.org/

[5] http://www.matecat.com/wp-content/uploads/2013/01/MateCat-D4.1-V1.1_final.pdf

[6] http://blogs.msdn.com/b/translation/archive/2010/03/15/collaborative-translations-announcing-the-next-version-of-microsoft-translator-technology-v2-apis-and-widget.aspx

[7] http://msdn.microsoft.com/en-us/library/hh847650.aspx

tion related to the post-edited actions (such as post-editing time) to be retrieved.

3 Design choices

Unlike other tools, the ACCEPT environment is divided into four distinct parts: a database, a server-side Application Programming Interface (API), a Web-based application (portal) and a client application where the actual post-editing work can be performed. While the client application is also Web-based, it differs from the Web-based portal because it is written in JavaScript (using JQuery libraries). This means that the client application can be integrated into any third-party Web application without requiring users to log on to the ACCEPT portal. This approach presents the advantage of bringing users closer to the actual content that should be post-edited.

The server-side API is used to create projects and tasks and upload and download data thanks to forms that are made available to specific users of the Web-based portal, known as project administrators (i.e. users that have sufficient rights to create projects). The API is also used to save any post-editing action that takes place in the client application. It is difficult to identify where machine translation fails from a post-editing perspective. By storing the post-editing actions (corrections of machine translated output and UI interactions) taken by multiple users in a central online location (the database), weaknesses of the MT systems can be quickly identified, especially since this information can be made available to project administrators in real-time. A crucial characteristic of the the ACCEPT environment is related to the way user activity is recorded (e.g. time spent, keys pressed): the recording is purely limited to what happens within the client application. This means that any activity performed outside of the ACCEPT environment (e.g. looking up a word definition or frequency using a search engine) will be ignored to respect the user's privacy. Further architecture and data management details are provided in the next sections.

3.1 Managing projects

The ACCEPT environment allows project administrators to create and manage post-editing projects. It is expected that project administrators have some community management expertise or responsibilities in order to create projects in which their community members may be interested in participating. It is the responsibility of the project administrators to decide who to invite to work on a project. The content that project administrators may upload for post-editing may range from short community contributions (such as forum or blog posts) to longer organisation-related documents such as training material (say, produced by a non-governmental organisation). Projects can be created to collect user edits and to possibly study those edits in detail thanks to the additional information that gets recorded during a post-editing session (such as time spent, types of keys pressed, etc.). This information can then be used by project administrators to identify participants who have contributed the most translations and/or the best translation quality to a project with a view to rewarding them using existing community rewarding schemes. In order to cater for multiple post-editing scenarios, the actual functionality of the client application is defined during project creation. To provide project administrators with as much flexibility as possible, the following interface configuration options are available during project creation:

- How is the post-editing task going to be conducted? In a monolingual or bilingual manner? If it is conducted in a bilingual manner, source segments may be viewed by users. Otherwise, source segments may be hidden. Default post-editing guidelines are also affected by this choice. A specific setting allows project administrators to decide whether users can override this project-level configuration (e.g. while a project may be configured in a monolingual manner, users can still view the source if they want to by clicking on a switch).

- Should translation options be used during a project? When this option is selected, alternative translation options provided by the MT system may be displayed to users.

- Should a specific feedback question be active? If so, which question/values should appear? When this field is used, the first option appears as the title of a drop-down list, followed by possible values. This allows project administrators to elicit feedback from users

on various aspects of the tasks (e.g. quality of the source, quality of the MT output, etc.)

- Should a post-task question be active? If so, which question should appear? Should a post-project link to a survey be present?
- What is the language pair of the project? If the language pair is English >French, the User Interface will be displayed in French. Currently, the following User Interface and project languages (i.e. languages in which content can be post-edited) are supported, but more could be easily added in the future: English, French and German. This selection also influences the language resources used by the spelling and grammar checker.
- Which user(s) should be invited to take part in this project and what text should be used to invite users to the project?

Once invited users have registered with the ACCEPT portal, they are presented with a list of tasks to work on. Once these tasks are completed, they disappear from their project page. Users can start working on a task by clicking its task ID. Once the task ID is clicked, the post-editing window appears based on the configuration that was specified by the project administrator. All user actions are saved automatically (e.g. segment-level comments, segment changes), but users are able to use the Undo and Redo functionality when editing segments. Users can close the window even if a task is not completed. This stops the global time count until the window is opened again. A task can be closed for good at any moment by the user (after being prompted to confirm their choice).

Once a project is created, data that should be post-edited by human reviewers can be imported into the system, as described in the next section.

3.2 Uploading data

The data format that is currently supported is based on a simple JSON format,[8] which must contain the following data:

- *text_id*: a string corresponding to a unique string identifier (e.g. hash value of the source text). A file with a text_id that is already in use in a given project can not be uploaded.
- *src_sentences*: an array of objects, which are pairs of sentences, in the form: *"text": "sentence"* (where sentence is a string)
- *tgt_sentences*: an array of objects, each containing a sentence pair, in the form: *"text": "sentence"*. Each object may also contain an optional options pair, in the form: *"options": option_array* (where *option_array* is an array of objects). Each object in the *option_array* should contain a pair of tokens (in the form *"context": "token"*, where *token* corresponds to a substring from a target sentence) and a *values* pair which should contain an array of objects. Each of these objects should contain an alternative phrase pair, in the form *"phrase": "token"*.

The number of objects in *src_sentences* must be equal to the number of objects in *tgt_sentences*. The final JSON format may also contain metadata, such as a contact email address, an MT tool name and tool ID, a source language code and a target language code.

The final JSON format may also contain an optional *tgt_templates* array to influence the display of the target sentences in the post-editing window. Templates were developed to define the layout of the post-editing tasks and how they are displayed in the editor. Standard DIV elements may be included within the *tgt_templates* array and each DIV element may contain any CSS style information within or around it. Each DIV corresponds to one segment in the editor, which means that the display of a paragraph can be defined individually. Each DIV element must include a *@TARGET@* sequence, which defines the position of the respective target sentence in the target text. The number of segments in *tgt_templates* needs to match the number of segments in *src_ sentences* and in *tgt_sentences* in order for the templates to be mapped correctly to the segments to be edited. To build a paragraph, for example, the sequence *style="display:inline"* can be included in front of the placeholder for the actual segment. Special characters must be encoded (e.g. < as *%3Cdiv*) since the server only accepts characters inside the ASCII character set.[9] An example of such a DIV is shown below:

[8] http://www.json.org/

[9] http://www.w3schools.com/tags/ref_urlencode.asp

"tgt_templates": [{ "markup": "%3Cdiv style=\"display%3Ainline;\"%3E@TARGET@ %3C/div%3E%3E " },
{ "markup": ... }],

When no template is specified, the default behaviour is adopted and displayed, which means that the post-editing task on the left of the editor contains no paragraph. The default style for a DIV is *style="display:block"*.

3.3 Designing the client application

While the actual functionality available in the client application is described in section 4, the present section focuses on technical design choices. When designing the client application of the ACCEPT post-editing environment, the focus was on developing a portable application purely written in JavaScript instead of relying on a standard Web application. A standalone Web application would have required users of an existing community to switch environments to perform post-editing tasks. The jQuery library was selected to build this portable application as a plug-in.[10] This library was selected because it is fast and widely used. It also has useful utility functions, good documentation and a large community. It offers cross-browser compatibility and full support for CSS3 selector specification. When it is used in conjunction with the jQuery UI library, CSS styles can be inherited and easily changed. [11] The logic around the technical aspect of the plug-in is based on collecting information from an HTML DOM object that self-configures the plug-in.

3.4 Recording post-editing actions

In order to assess whether MT or post-editing was effective, analysing the actions performed or the time spent by a given user on a given task may be extremely informative. The ACCEPT environment records actions and time in the following manner.

1. When a user clicks on any task link, the post-editing window opens.

2. When the post-editing window opens, the global time count for the task starts. This action is captured in a phase called *start_pe*.

3. When a user clicks on any segment in the target text (say, segment 1), the target segment appears in the editing window.

4. When the user takes any action in the editing window, a time-stamp is recorded for the first revision of the current segment. This action is recorded in a phase called *r1.1* (where *r* stands for revision, *1* for segment 1, and *1* for revision 1).

5. The user now leaves the current segment, by clicking on another segment (say segment 3), and starts editing it. There is a new time-stamp for this new segment, at revision 1. Keystrokes are recorded again.

6. The user is not satisfied with segment 1, and clicks on segment 1. There is a new time-stamp for segment 1 at revision 2. Editing actions are recorded.

7. The user clicks on segment 2. No further actions are taken by the user. No extra information is recorded.

8. The user is satisfied with the result of this task, clicks the *Complete Task* button, and is asked to confirm. If the confirmation is positive, the status for the document is *FINISHED*. In this case, the task disappears from the overview page and the global time count for the task stops. Otherwise, the status of the task is *UNFINISHED*. In that case, the task will still be displayed on the project page. The global time count for the task stops until the task is re-opened.

During the post-editing process, users often behave in unforeseen ways or come across unforeseen issues. In most existing solutions, user actions and interactions can only be investigated once the post-editing process has been completed. The ACCEPT environment provides instant access to user progress, so that project administrators can identify potential problems and solve them on the spot without losing valuable time and data. This real-time recording of the users' progress facilitates a more efficient project management which can also be used for rewarding users according to their progress automatically. The next section focuses on the format of the report used to export user activity.

[10] http://jquery.com/
[11] http://jqueryui.com/

3.5 Generating activity reports

Actions and interactions from multiple users are collected and grouped together in real-time. A standard format (XLIFF) is used for the export of information, thus maximising interoperability.[12] Figure 1 presents the mapping used to group a data point (such as the number of arrow keys pressed in a revision of a given segment) in an XLIFF *count* element with a *count-type* attribute whose value has a custom value, *x-arrow-keys*.

As shown in Table 1, multiple XLIFF elements are used to represent post-editing activity data. The actual target text entered by a user when modifying a machine-translated segment is present in the *body* element in *target* elements of *trans-unit* elements. Each revision is saved as a distinct phase labeled with a unique *phase-name* attribute. This attribute value can then be cross-referenced with a *count* element and a *phase* element that have the same attribute value. These *count* and *phase* elements contain metadata information about the activity performed by the user. For instance, the number of arrow keys pressed or any comment the user may have made on the quality of the original machine-translated segment can be represented using these elements.

Project administrators can then export post-editing activity data at the user-, document- or project -level by generating XLIFF files, such as the one shown in Figure 1:

Figure 1 shows the activity one user performed on one task. A task is mapped to an XLIFF *file* element, which contains a *header* element and a *body* element. The *body* element is used to capture all of the texts that were used and produced during the post-editing process, including the source text, the original machine-translated text and any revision that may have been produced. The *header* element contains detailed information on the actual actions that occurred during the post-editing process (e.g. types of keys being pressed, number of times the text checker was used, alternative options used, etc.).

```
<xliff version="1.2">
<file original="123"
    source-language="en" target-language="fr" datatype="text"
    category="IT" product-name="TEST">
<header>
    <phase-group>
    <phase phase-name="mt_baseline" process-name="Machine Translation"
        tool="sb" toold-id="mydemoMT"
        date="2013-05-03T14:02:40.470235" contact-email="11850@test.com"/>
    <phase phase-name="start_pe" process-name="bilingual"
        tool="ACCEPT Portal" toold-id="ACCEPT Post Edit Plug-in 1.0"
        date="2013-05-03T13:17:22" contact-email=""/>
    <phase phase-name="r1.1" date="6/6/2013 1:11:47 PM"
        contact-email="test@test.com"/>
    </phase-group>
    <count-group name="1">
    <count phase-name="r1.1" count-type="x-keys" unit="instance">1</count>
    <count phase-name="r1.1" count-type="x-delete-keys" unit="instance">1</count>
    <count phase-name="r1.1" count-type="x-editing-time" unit="x-seconds">2.922</count>
    <count phase-name="r1.1" count-type="x-typing-time" unit="x-seconds">2.922</count>
    </count-group>
    <count-group name="2"/>
</header>
<body>
    <trans-unit id="1">
        <source>This is a new test</source>
        <target phase-name="r1.1">C'est un nouveau test</target>
        <alt-trans phase-name="mt_baseline">C'est une nouvelle test</alt-trans>
    </trans-unit>
    <trans-unit id="2">
        <source>the old one was very old</source>
        <target phase-name="mt_baseline">L'ancien a été très vieux</target>
    </trans-unit>
</body>
</file>
</xliff>
```

Figure 1: Report in XLIFF format

Figure 1 shows the activity one user performed on one task. A task is mapped to an XLIFF *file* element, which contains a *header* element and a *body* element. The *body* element is used to capture all of the texts that were used and produced during the post-editing process, including the source text, the original machine-translated text and any revision that may have been produced. The *header* element contains detailed information on the actual actions that occurred during the post-editing process (e.g. types of keys being pressed, number of times the text checker was used, alternative options used, etc.).

4 Client Functionality

The guiding principle followed when selecting functionality for the plug-in was that the user interface should be as simple as possible, using few but well-known button icons. One of the first questions we had to answer when creating the plug-in was how the text should be displayed to users. It was felt that displaying text in a grid format would be intimidating to non-professional translators. Instead, we decided to present the whole target text to edit in a column (left), thus giving users the ability to navigate from one sentence to another by clicking on individual sentences. The editing takes place in a separate column (right), segment by segment, as shown in Figure 2:

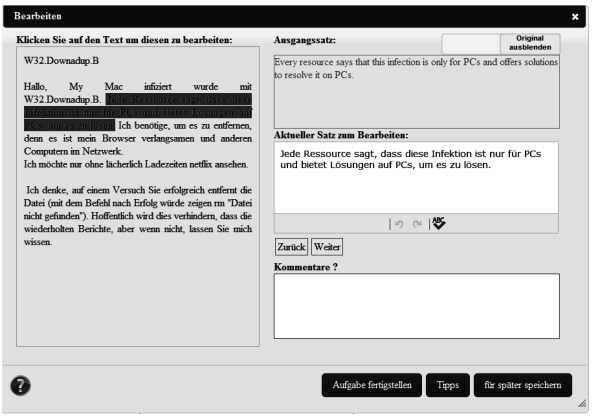

Figure 2: Client application

Figure 2 shows that the ACCEPT plug-in is target text-driven since the objective of the task is to post-edit pre-translated content. Users are invited to make edits to an existing target text rather than creating a target text from scratch by getting some

[12] http://docs.oasis-open.org/xliff/xliff-core/xliff-core.html

Description	XLIFF count Type	Unit	Level	Example
Total time (active editing window)	x-total-time (in phase-name="complete_pe")	Seconds	Task	
Total editing time (all revision-level editing times)	x-editing-time (in phase-name="complete_pe")	Seconds	Task	
Number of keys pressed	x-keys	Instance	Revision	
Number of space keys pressed	x-white-keys	Instance	Revision	
Number of alphanumerical and non-white keys pressed	x-nonwhite-keys	Instance	Revision	a,J,{
Number of arrow keys pressed	x-arrow-keys	Instance	Revision	$\uparrow, \leftarrow, End, Home$
Number of other keys pressed	x-other-keys	Instance	Revision	CTRL,Shift
Number of triggered checks	x-checking-usage	Instance	Revision	
Selection of alternative translation options	x-trans-options-usage	Instance	Revision	
Time spent editing	x-editing-time	Seconds	Revision	
Description	**XLIFF Mapping**	**Type**	**Level**	**Example**
Used Translation Options	note annotates="target" from="trans_options"	String	Revision	Hat |||Ist |||0
Generic Comment	note annotates="general" from="user"	String	Revision	Easy to edit!
Custom Comment	note annotates="target" from="user"	String	Revision	Terminology
Text entered	target of alt-trans if not final or target of trans-unit if final	String	Revision	Ist jemandem das schon mal begegnet?
Global Comment	note annotates="general"	String	Task	Easy!

Table 1: XLIFF elements used to display PE actions

inspiration from a source text. This choice is motivated by the fact that target users of this application are members of a community who may have a very limited knowledge of the source language. To fully support this use case, the user interface of the plug-in has already been fully localised from English into French and German, and more languages will be easily added in the future.

4.1 Displaying alternative translation options

The best output of an MT system (especially the output of an SMT system) corresponds to a product of choices between multiple options, but some of these options sometimes turn out to be sub-optimal in a given context. While experienced post-editors can easily correct these poor choices by selecting more suitable morphological, lexical or syntactic options without being prompted, it has been shown that lesser-equipped translators can benefit from having access to alternative translation options (Koehn, 2010). As described in section 3, the JSON format used to upload data into a given project may contain such alternative options. The ACCEPT plug-in currently expects each alternative translation option to be present in an array associated with each target token. Once a token has been identified by a user (by hovering over it and clicking it), its alternative options are displayed in a contextual list, where each item can be selected in order to replace the original token in the target text. The use of this functionality is recorded in the XLIFF report using a *count* element with a *count-type* attribute whose value is *x-trans-options-usage*. The options that were actually selected by the user are recorded in *note* elements.

4.2 Checking Target Content

The output of MT systems is sometimes ungrammatical, so users may benefit from some assistance to identify those parts of the output that should be modified. Spelling and grammar checking is there-

fore made available by embedding a pre-editing plug-in in the post-editing plug-in itself. This functionality can be triggered by clicking the *ABC* icon. Once this icon is clicked, a pop-up window appears and misspelt words or ungrammatical phrases are underlined. The user can then select a suggestion or ignore the recommendation provided by the tool. Supported languages currently include English, French and German. It is currently not clear how useful the default rules are for checking MT output, but they can also be used once the MT output has been post-edited. The use of this functionality is recorded in the XLIFF report using a *count* element with a *count-type* attribute whose value is *x-checking-usage*.

4.3 Showing the source

Post-editing is traditionally believed to be most successful in a bilingual manner (i.e. post-editing with reference to the source text) for the reason that meaning that may have been lost/distorted in the machine translation process can be retrieved from the source text. While research in monolingual post-editing is scarce, especially in regards to domain experts as post-editors rather than linguists/translators, providing the post-editor with the opportunity of choosing the post-editing set-up dynamically (i.e. monolingual/bilingual) has been identified as a potential way of minimising or preventing user frustration. This is supported by feedback that has been gathered in internal studies, which indicated that users were eager to see the source, as further described in Section 5.

To illustrate switching between bilingual and monolingual modes, consider what happens if the project default is the monolingual mode. The source will then not be shown in the interface when a task is opened initially. The user can then decide to switch to being shown the original segment for the current segment. Regardless of how many switches are performed per segment, the last state the switch is in is retained for the next segment a user chooses to edit. It can be switched at any time. When the editor is closed, the page is refreshed or a new task is selected, the project default is displayed again (in this case the source is not shown).

5 Evaluation

A pilot study with 8 participants was recently conducted with a view to analyse the types of edits made by volunteer post-editors (most of whom were forum users with no formal translation or post-editing experience). During this study, we found that the interface was not straight-forward to use because it differed from editors users were familiar with (e.g. MS Word). This resulted in users copying and pasting content into other editors or editing all content in one segment. In addition, the help button was not visible enough and the instructions were not clear enough. These issues were addressed by making the help button more visible and by creating a short training video for future users, consisting of a screen recording and a voice-over in the participants' native language. Since these improvements have been made, a new study has been conducted with 18 participants (five of whom had translation experience), and it has revealed that there is no longer any confusion on how to use the interface. We also took this opportunity to ask users three specific questions about the interface:

- Which feature did you like best?
- Which feature did you like least?
- Which feature did you miss most?

The results presented in Table 2 show that most users were satisfied with the feature set of the ACCEPT environment. When asked to identify the feature they liked least, half of the respondents answered "None", suggesting that everything was working as expecting. Based on the feedback received, areas to improve include a more comfortable display of the target text (without any scrolling) and the ability to have access to the whole source text.

The lukewarm feedback received in relation to the spell-checking feature confirms the need to use a tool that has been specifically designed with machine translation output and post-editing in mind. Traditional spelling and grammar checkers are usually not trained on machine translation output, so they can generate false alarms when used in a post-editing context. Some of the enhancement suggestions also reveal that experienced users (e.g. people with translation expertise) are interested in having access to features found in existing translation environments (such as advanced editing tools or online dictionaries). It is also worth highlighting two other user suggestions: having to ability to

Best	%	Worst	%	Missed	%
Show source	77.8	None	50	None	50
All	11.2	Spell checker	27.8	Dictionary/thesaurus/alternative translations	10
Yellow text highlighting	5.5	"Next" button	11.2	Show whole source text	5
Whole text on left side	5.5	Small fonts	5.5	No scrolling	5
		Vanishing tasks	5.5	Show how others have translated this	5
				Revert to MT segment	5
				Use proofreading symbols	5
				Show editing tool as menu bar	5
				Display statistics and badges	5
				Have another window for draft sentences	5

Table 2: Usability survey results

revert to the original MT output and being able to see how other users may have post-edited the same segment.

6 Conclusions and future work

This paper has presented a simple, yet powerful, novel post-editing environment aimed at community users who may have very limited translation expertise. We described the architectural design choices that were made to create a flexible environment that clearly segregates the client application from the rest of the environment. User actions and interactions that take place in the client application are captured via the ACCEPT API, stored in a database, and made available in real-time to project administrators via a downloadable report based on the XLIFF format. Future work will focus on documenting the API of the ACCEPT environment, so that post-editing tasks can be created easily, without necessarily having to upload input files manually. An extension of this work will also include necessary changes to allow the use of the client application outside of the ACCEPT portal. We have already made some progress in this area by allowing certain users to make use of the ACCEPT client application inside Amazon Mechanical Turk' HITs.[13] We will also investigate the possibility to integrate functionality provided by advanced (S)MT systems, such as the mining of complete search graphs to display useful alternative translation options. Finally, we would also like to conduct an evaluation comparing the usage of this environment with the usage of another existing tool to benchmark how long post-editing takes in each environment.

[13] https://www.mturk.com

Acknowledgements

The work presented in this paper is being supported by the European Commission's Seventh Framework Programme (Grant 288769). The authors would like to thank Fred Hollowood and Jason Rickard for their insights during the initial design phase, all of the users of the ACCEPT post-editing environment who have provided some feedback to date, as well as the reviewers of this paper for their comments.

References

Wilker Aziz, Sheila Castilho, and Lucia Specia. PET: a Tool for Post-editing and Assessing Machine Translation. In Nicoletta Calzolari, Khalid Choukri, Thierry Declerck, Mehmet Ugur Dogan, Bente Maegaard, Joseph Mariani, Jan Odijk, and Stelios Piperidis, editors, *LREC*, pages 3982–3987. European Language Resources Association (ELRA), 2012.

Michael Carl. Translog-II: a Program for Recording User Activity Data for Empirical Reading and Writing Research. In *LREC*, pages 4108–4112, 2012.

Giselle de Almeida and Sharon O'Brien. Analysing Post-Editing Performance: Cor-

relations with Years of Translation Experience. In François Yvon and Viggo Hansen, editors, *EAMT 2010, 14th Annual Conference of the European Association for Machine Translation*, Saint-Raphaël, France, 2010.

Michael Denkowski and Alon Lavie. TransCenter: Web-Based Translation Research Suite. In *AMTA Workshop on Post-Editing Technology and Practice Demo Session*, San Diego, CA, 2012.

Jakob Elming and Ragnar Bonk. The CASMACAT workbench: a tool for investigating the integration of technology in translation. In *Proceedings of the International Workshop on Expertise in Translation and Post-editing - Research and Application*, Copenhagen, Denmark, 2012.

Raymond Flournoy and Christine Duran. Machine Translation and Document Localization Production at Adobe: From Pilot to Production. In *MT Summit XII: proceedings of the twelfth Machine Translation Summit*, 2009.

Spence Green, Jeffrey Heer, and Christopher D Manning. The efficacy of human post-editing for language translation. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '13, pages 439–448, New York, NY, USA, 2013. ACM.

Ana Guerberof Arenas. *Productivity and quality in the post-editing of outputs from translation memories and machine translation*. PhD thesis, Universitat Rovira i Virgili, Spain, 2012.

Philipp Koehn. Enabling Monolingual Translators: Post-Editing vs. Options. In *Proceedings of Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 537–545, Los Angeles, California, 2010.

A Kumaran, K Saravanan, and Sandor Maurice. wikiBABEL: community creation of multilingual data. In *Proceedings of the 4th International Symposium on Wikis*, WikiSym '08, pages 14:1—-14:11, New York, NY, USA, 2008. ACM.

Linda Mitchell and Johann Roturier. Evaluation of Machine-Translated User Generated Content : A pilot study based on User Ratings. In *Proceedings of EAMT 2012*, pages 61–64, Trento, Italy, 2012.

Mirko Plitt and François Masselot. A Productivity Test of Statistical Machine Translation Post-Editing in a Typical Localisation Context. *Prague Bull. Math. Linguistics*, 93:7–16, 2010.

Johann Roturier and Anthony Bensadoun. Evaluation of MT Systems to Translate User Generated Content. In *Proceedings of the Thirteenth Machine Translation Summit*, pages 244–251, Xiamen, China, 2011.