Collaborative Web UI Localization, or How to Build Feature-rich Multilingual Datasets

Vicent Alabau
PRHLT Research Center
Universitat Politècnica de València
valabau@prhlt.upv.es

Luis A. Leiva
PRHLT Research Center
Universitat Politècnica de València
luileito@prhlt.upv.es

Abstract

We present a method to generate feature-rich multilingual parallel datasets for machine translation systems, including e.g. type of widget, user's locale, or geolocation. To support this argument, we have developed a bookmarklet that instruments arbitrary websites so that casual end users can modify their texts on demand. After surveying 52 users, we conclude that people is leaned toward using this method in lieu of other comparable alternatives. We validate our prototype in a controlled study with 10 users, showing that language resources can be easily generated.

1 Introduction

Today most websites are looking forward to making their contents available in more than one language, mainly to reach a global audience, to gain a competitive advantage, or just because of legal requirements. To this end, adapting user interface (UI) texts through translation—or "localization"—is a central task, since its result affects system usability and acceptability. Actually, translation is just one of the activities of localization yet the most important overall (Keniston, 1997).

Recently there have been significant improvements in machine translation (MT) technology, to the extent that, in particular contexts such as medical prescriptions or knowledge-base articles, machine-translated content is qualitatively comparable to that of human-translated (Dillinger and Laurie, 2009). However, for MT systems to excel at UI localization not only it is needed an important amount of training data, but also the data must be especially tailored to the particularities of UI messages. Indeed, translating the text in an interface is a challenging task, even for trained human translators (Muntés-Mulero et al., 2012).

Parallel data offer a rich source of additional knowledge about language, and a sound basis for both translation and contrastive studies (McEnery and Xiao, 2007). Although there are some valuable tools to build multilingual parallel corpora, they are still limited when it comes to the exploitation of UI-based resources. Thus, we propose a novel approach: delegating the corpus generation to the end users of software applications, as a result of a regular interaction with such applications. To support our approach, we developed a proof-of-concept web-based prototype, motivated by the fact that nowadays people use web browsers more than any other class of software. Moreover, software translation poses two interesting challenges: *1)* user interface (UI) strings appear anywhere in the developer's language of choice whereas content is typically generated and consumed in the user's language; *2)* UI bilingual sentences can be enriched with metadata to handle disambiguation.

2 Related Work

In the past, several methods have been developed to build parallel corpora by automatic means, e.g., by mining Wikipedia (Smith et al., 2010), web pages with a similar structure (Resnik and Smith, 2003), parliament proceedings (Koehn, 2005), or using specialized tools such as OPUS (Tiedemann, 2012). However, in the end, parallel texts are scarce resources, limited in size and language coverage (Munteanu and Marcu, 2005).

In addition, many tools such as Crowdin[1], SmartLing[2], and Launchpad[3] do support collaborative translation. However, for these tools to work properly, applications must be internationalized beforehand. Besides, Google Translator Toolkit[4] allows contributing with translations. However,

© 2014 The authors. This article is licensed under a Creative Commons 3.0 licence, no derivative works, attribution, CC-BY-ND.

[1]**http://www.crowdin.net**
[2]**http://www.smartling.com**
[3]**http://translations.launchpad.net**
[4]**http://translate.google.com/toolkit/**

the proposed translations are not rendered on the web page unless one uses the Website Translator tool *and* owns the site. Furthermore, it is oriented to translating *content* and not UI elements such as buttons, drop-down lists, etc. that otherwise may carry valuable language information.

Probably, the closest work in soul to ours is Duolingo (von Ahn, 2013), an effort to collaboratively translate the Web while users are learning a language. However, we are interested in providing computer users with a means of editing the text of any website on demand, only when it is needed.

More importantly, current tools force users to switch and use said tools, which may prevent them from contributing. Also, user contributions are not shown until the application owner decides to do so, thus hindering collaboration. Therefore, we feel another collaborative translation method is needed.

3 User Survey

We prepared a 2-question survey in order to identify to what extent would users be motivated to translate or edit translations in a computer application or a website. The first question (**Q1**) asked the preference degree to using 4 different methods:

1. **M1**: Editing the application source code.
2. **M2**: Installing a dedicated tool.
3. **M3**: The application features a menu option.
4. **M4**: Editing text in-place, at runtime.

The second question (**Q2**) asked the willingness to personalize the texts displayed in an application, provided that there were an easy method to do it. We included example images for each instance case, and answers to **Q1** were randomly presented to the users, to avoid possible biases. Both questions were scored in a 1–5 Likert scale (1: strongly disagree, 5: strongly agree). The survey was then released online via Twitter, Facebook, and word-of-mouth communication. Eventually, 52 users (24 females) aged 19–34 from 5 countries (USA, UK, France, Spain, and Germany) participated in the survey. The results are shown in Table 1.

	M1	M2	M3	M4	Q2
M	1.79	2.37	3.27	4.58	4.27
Mdn	2	2	3	5	4
SD	0.98	0.97	0.95	0.82	0.88

Table 1: Detailed survey results.

As observed, a preference for in-place runtime translation (**M4**) is evident over the rest of the considered options. Installing dedicated software (**M2**) is not seen as a likable approach, and even less editing the source code of the application (**M1**). On the other hand, having a translation facility bundled with the application (**M3**) is a significant enhancement. This is somewhat already implemented in most Linux programs, e.g., the official GNOME image viewer,which allows users to seamlessly collaborate worldwide to translate the program. Nevertheless, as previously pointed out, **M4** seems to be the most comfortable option.

Regarding the willingness to personalize texts (**Q2**), as expected, people are favorably predisposed to do so if they were given an easy-to-use method such as the one we are proposing. Together with the previous answers, this survey reveals that our method would allow regular computer users to (indirectly) contribute with translations. This suggests in turn that occasional users of an application or arbitrary visitors of a website are more likely to submit a translation pair, which would dramatically facilitate corpus construction, both in terms of human effort and time.

4 Method Overview

Apparently, users are eager to contribute with translations when they can instantaneously personalize their applications and the collaboration effort has a low entry cost. Thus, we propose a method were translations are carried out *just-in-time* and *in-place*. First, just-in-time implies that a translation takes place at the very same moment that the user needs it. For instance, when a user spots a sentence that has not been translated into her language, or a translation error is bothering her, she is simply able to amend the text on the UI. Second, in-place editing means that translation is performed on the same UI, not in another application, so that the overhead introduced by task switching has minimal impact. This localization strategy has shown some advantages over more traditional methods (Leiva and Alabau, 2014).

The core idea of our method is adapting the behavior of UI widgets so that they can switch to an *edit mode* when some accelerator is used. Note that the application should work as it was originally designed, however the behavior of the widgets would change only on demand (see Figure 1). While in theory this could be incorporated to any major UI library (e.g., Qt, GTK, MFC, Co-

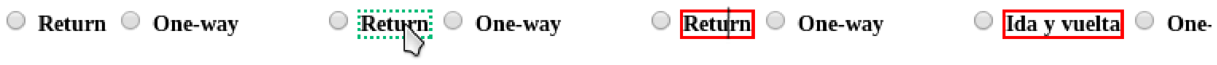


Figure 1: Example of *edit mode*. While `CTRL` is pressed, elements are highlighted as the mouse hovers them. Then, the user clicks on the element, which becomes editable, in order to change its content.

coa), in this paper we test a method that is suitable for web-based UIs. For simplicity, the method is deployed as a bookmarklet (no installation, just drag-and-drop, available for all browsers), which is more compatible than using extensions or plugins. The method can be roughly summarized as follows: *1)* a welcome menu is shown when clicking on the bookmarklet; *2)* resource strings are automatically extracted in the original language from text nodes, `alt` attributes, `form` elements, etc. along with a unique identifier (XPath); *3)* user's previous translations, if any, are loaded and applied to the UI; *4)* event listeners to receive user interaction are attached to UI elements; *5)* when the user activates the *edit mode*, UI elements become *content-editable* items, or a modal window pops up as a fallback mechanism; *6)* user information is collected, such as locale, geolocation by IP, etc. *7)* finally, the user can submit her contributions by clicking again on the bookmarklet.

5 Evaluation

We performed a controlled evaluation to assess if our method was worth being deployed at a larger scale. Thus, we recruited 10 Spanish users with an advanced English level. Participants were told to translate while interacting with a small airline website (5 pages) and one section of the popular Wordpress platform. At the end of the session, users submitted their translations to our server.

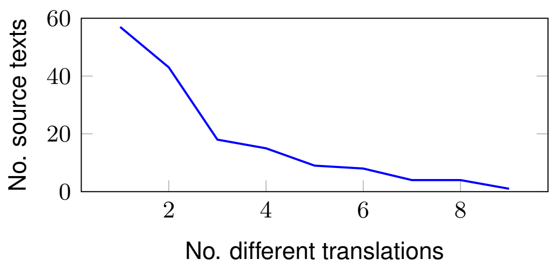
In 5 minutes, 159 out of the 205 potentially translatable sentences were identified by the users. On average, each user contributed with 114 (*SD*=4) sentences. Not all sentences were translated because some of them only appear under special circumstances like error messages or hidden options in menus, whereas others have low saliency (e.g., a copyright notice). Figure 2a shows the histogram of sources with different translations. It can be observed that more than a half of the sources received multiple translations, while it was not unusual to have up to 4 different translations for each source. Conversely, Figure 2b shows the histogram of the number of times the most voted translation was indeed produced by the agreement of n users. It turns out that users showed full disagreement only on 24 sentences. For the other sentences, at least two users agreed at any time. In addition, we can see a peak when 9 and 10 users agreed. This is explained in part because some sources were fairly simple to translate (such as navigation links) and thus it was expected that users would submit similar translations.

In general, users reported that they were happy to test our method for translating web pages. They felt the technique was easy to use, and expressed an intention to contribute with translations for their favorite applications. Hence, it seems plausible that a larger scale deployment would be successful.

6 General Discussion

Our method allows users to achieve an immediate benefit, since the website is being adapted to their language needs as they contribute to translating (and personalizing) it. At the same time, researchers also benefit from these contributions, since valuable language resources are being generated in the long run. Further, the method leads to having multiple references for a given source text, coming from different users worldwide, which allows for better training and evaluation of MT systems. More importantly, resources are ultimately supervised by humans—which provides valuable ground truth data—and can be deployed for potentially *any* language. Last but not least, our method enables "contextualized translation", in the sense that additional metadata are coupled to the traditional source-target language pairs, such as the type of widget (e.g., button, label, etc.), geolocation, locale, or the user agent string.

The survey gave us intuition regarding whether regular users would engage to contribute with casual translations. Nevertheless, as in any collaborative tool, the user needs a motivation to carry out any task. We believe that our proposal adds great value to how users experience computer software since, right from the beginning, they can fix translation errors and personalize their favorite applications. In contrast to other approaches where



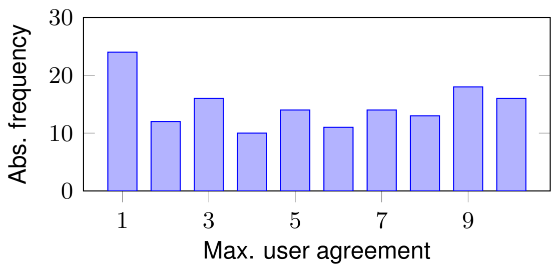


Figure 2: Distribution of different translations per source (2a) and histogram of user agreements (2b).

the user contributions are used to merely collect data, here these contributions are rendered immediately on the UI, so the benefit becomes instantaneous. Besides, as more and more data are collected, they can be used to initially populate a web page or application with the consensus translations from other users. This is especially interesting for minority languages, where a few users with knowledge of said minority language can make the UI accessible to the rest of users. Also, information reported by the browser can provide translations tailored to the user context, e.g., country or operating system. Hopefully, the low entry cost of our approach will reduce the burden on the user and thus foster collaboration.

In addition, the language resources that our method is able to collect provide unprecedented value for the MT community. First, potentially any language with a representative user base can generate parallel data. What is more, sentence pairs are properly aligned, since they come from the very same UI element, and multiple references may be available. Furthermore, translations are performed with a visual context. Thus, not only the chances that translations are appropriate will improve, but also language resources can be tagged with feature-rich metadata. For instance, the type of UI element (e.g., paragraph, button, link) or the text of a header or a label that relates to it, all can be used as additional information to provide better disambiguation in MT (Muntés-Mulero et al., 2012). Even so, personal information—if available and always under the user consent—can provide resources for adaptation of general models to specific dialects, or to target different age groups.

Acknowledgments

Work supported by the FP7 of the European Commission under grant agreements 287576 (CASMACAT) and 600707 (tranScriptorium).

References

Dillinger, M. and G. Laurie. 2009. Success with machine translation: automating knowledge-base translation. ClientSide News.

Keniston, Kenneth. 1997. Software Localization: Notes on Technology and Culture. Working Paper #26, Massachusetts Institute of Technology.

Koehn, Philipp. 2005. Europarl: A Parallel Corpus for Statistical Machine Translation. In *Proc. MT Summit*, pages 79–86.

Leiva, Luis A. and Vicent Alabau. 2014. The impact of visual contextualization on ui localization. In *Proc. CHI*, pages 3739–3742.

McEnery, Anthony and Zhonghua Xiao, 2007. *Incorporating Corpora: Translation and the Linguist*, chapter Parallel and comparable corpora: What are they up to? Multilingual Matters.

Munteanu, Dragos Stefan and Daniel Marcu. 2005. Improving Machine Translation Performance by Exploiting Non-Parallel Corpora. *Computational Linguistics*, 31(4):477–504.

Muntés-Mulero, V., P. Paladini Adell, C. España-Bonet, and L. Màrquez. 2012. Context-Aware Machine Translation for Software Localization. In *Proc. EAMT*, pages 77–80.

Resnik, Philip and Noah A. Smith. 2003. The Web as a parallel corpus. *Computational Linguistics*, 29(3):349–380.

Smith, Jason R., Chris Quirk, and Kristina Toutanova. 2010. Extracting parallel sentences from comparable corpora using document level alignment. In *Proc. NAACL*, pages 403–411.

Tiedemann, Jörg. 2012. Parallel Data, Tools and Interfaces in OPUS. In *Proc. LREC*, pages 2214–2218.

von Ahn, Luis. 2013. Duolingo: learn a language for free while helping to translate the web. In *Proc. IUI*, pages 1–2.