Towards Higher Quality Internal and Outside Multilingualization of Web Sites

Christian Boitet, Valérie Bellynck, Mathieu Mangeot, Carlos Ramisch

Towards Higher Quality Internal and Outside Multilingualization of Web Site

Christian Boitet, Valérie Bellynck, Mathieu Mangeot, Carlos Ramisch

GETALP (Groupe d'Étude pour la Traduction Automatique et le Traitement Automatisé des Langues et de la Parole)
LIG (Laboratoire d'Informatique de Grenoble, Grenoble Informatics Laboratory)
UJF (Université Joseph Fourier), UFR IMAG, France
{Christian.Boitet, Valerie Bellynck, Mathieu.Mangeot, Carlos.Ramish@imag.fr}

Abstract

The multilingualization of Web sites with high quality is increasingly important, but is unsolvable in most situations where internal quality certification is needed, and not solved in the majority of other situations. We demonstrate it by analyzing a variety of techniques to make the underlying software easily localizable and to manage the translation of textual content in the classical *internal mode,* that is by modifying the language-dependent *resources*. A new idea is that volunteer final users should be able to contribute to the improvement or even production of translated resources and content. For this, we have developed a PHP piece of code which naive webmasters (not computer scientists nor professional translators) can add to a Web site to enable *internal* multilingualization by users with enough access rights: in *management mode*, these users can edit the texts of titles, button labels, messages, etc. in text areas appearing *in context* in the Web page. If Web site developers follow some recommendations, all textual interface elements should be localizable in this way.

Another angle of attack, applicable in all cases where navigating a site though a *gateway* is possible, consists in replacing the problem of diffusion by the problem of access in multiple languages. We introduce the concept of iMAG (interactive Multilingual Access Gateway, dedicated to a Web site or domain) to solve the problem of *higher quality* multilingual access. First, by using available MT systems or by default morphological processors and bilingual dictionaries, any page of an *elected* website is made instantly accessible in many languages, with a generally low quality profile, as through usual translation gateways. Over time, the quality profile of textual GUI elements, Web pages and even documents (if accessible in html) will improve thanks to outside contributors, who will post-edit or produce the translations from the reading context. This is only possible because the iMAG associated to the website stores the translations in its translation memory (TM) and the contributed dictionary items it its dictionary. The TM has quality levels, according to the users' profiles, and scores within levels. An API will be proposed so that the developers of the *elected* website can connect their to its iMAG, retrieve the *best level* translations, certify them if necessary, and put them in their localized resources. At that point, external localization meets internal localization.

Keywords: Multilingual access, Machine Translation (MT), online MT, Web localization, collaborative translation environment, extended translation memory, translation of dynamic Web sites.

Introduction

More and more information is disseminated trough Web sites, and there is a desire or even an obligation to make it accessible in many languages, with a quality level far above what general-purpose MT can achieve. International bodies such as the EU or UNESCO or ITU have actually a legal obligation to do it (in more than 20 languages for Europe, in 6 languages for UNESCO and ITU). Large and small firms have to publish their documentation in many languages, and to adapt, or *localize* their products. Software publishers are a case in this point: IBM localizes in 25 languages, Adobe in 32, HP in 40, MicroSoft in 45, etc. Another kind of would be multilingual quality publishers are Web communities dedicated to causes such as human rights (PaxHumana), peace, health, or simply to some domains (e.g., cooking).

But official organizations legally bound to publish in many languages don't do it in reality, or only for a very small proportion of the information, even if they spend a lot on translation. Many firms can also not bear the costs of HQ translation, and limit their localization to a few languages, which reduces their potential market, and dramatically increases the cost of their hot lines, under heavy pressure because users cannot understand well enough the English documentation, or the low-quality translations provided.

We propose two complementary approaches to solve this problem. First, *outside* localization is possible for Web sites, although it may be differently handled for program-related textual material (labels of GUI elements such as titles, buttons, and program messages) and for content-related material. Whether that content is stored in static Web pages or dynamically generated from a database makes no difference in our approach, based on associating an *iMAG* (interactive Multilingual Access Gateway) to an *elected* Web site or domain.

Second, *inside* localization techniques can be dramatically improved in the case of Web sites, with respect to classical techniques.

The first step is to allow not only developers, but managers and reliable users of Web sites to edit the localized *resources* (program-related as well as

content-related) in their language in an easy way. According to the design of the Web site, the localized strings appear immediately, or only after the next build of the packages used.

The second step is to allow *localization in context*, which is a dream for translators of software packages. For the moment, our technique is limited to Web sites. If the developer follows some guidelines and organizes the code appropriately, the manager and reliable users can localize the Web site *in context*, while using the site, switching *seamlessly* between usage context and editing context: in *management mode,* an editable text area appears next to each localizable object, and the page is updated after the next click outside it.

The first section discusses in more details the differences between *inside* and *outside* localization techniques, according to various *translational situations*. The second section describes pros and cons of classical inside techniques, applied to Web sites, and introduces an innovative *in context* localization technique. The third section presents outside localization techniques, and introduces the iMAG concept, which extends the *translation gateway* concept to allow contributive quality improvement.

1 Inside and outside localization

The proper term to use in our context is *multilingualization* or *linguistic porting*, rather than *localization*, which implies not only translation into other languages, but more adaptations. For example, a vertical menu placed at the left for left to right scripts (e.g. English, Hindi) should be placed to the right in the case of right to left scripts (e.g. Arabic, or Urdu). We will however often use *localization* here instead of *multilingualization* for the sake of brevity.

Inside localization is done by working on the code and resources of an application (a Web site in particular), while *outside* localization is done by translating and adapting what is produced by the application, without any intervention on its code and resources.

1.1 An unsolvable problem: quick certified HQ multilingual diffusion

Many official organizations such as the European Union (EU) or UNESCO have several official languages in which they are supposed to disseminate all their documents. However, any *multilingualization index* of their websites is abysmally low. Here is an example taken from the EU site, where that index is less than 25% (6/23), even at the first level. When one goes down 2 clicks, it decreases to 4—8% (1—2/23).

That happens often because of the presence of a (understandable) *certification policy*. No material can be disseminated without formal approval. In the case of translation, that means that professional translators hired by the organism or firm have to "bless" any translation before it is published as coming from it.

Figure 1: low multilingualization index at EU site

In 2004-05, a group of 3 labs translated the textual material of the B@bel Web site of Unesco from English into Chinese, French, Russian, Spanish (and later Thai), 4 of the 5 other Unesco official languages. That material comprised 42301 words, about 169 standard pages, divided in 2896 segments of 14.9 words in average (the 1000 longest were 15.6 words long in average). The translations were all revised by native speakers with good knowledge of the domain.

Although this had been done in the framework of a research contract from Unesco, it was later impossible to get Unesco to include these translations in its database. That database is indeed prepared to handle the 6 languages in question, but remains almost empty for all but English.

1.2 Multilingual access with unguaranteed & unlabeled quality is not enough

Using free translation gateways is also not a solution when a really good understanding of the content is necessary. Trying to browse in French (or even worse in German) the Europe Web site(s) for calls for proposals is a conclusive experience. These calls should (by European law!) be available in all 23 official languages at the same time, with the same quality. The 1700 translators of the EU are certainly not enough to cope, and that number did not considerably augment with the number of languages growing from 9 around 1982 to 23 in 2007, 25 years later.

1.3 Misunderstandings about progress in MT

The kind of MT which would be necessary to cover that kind of need is simply not there. There are many claims about the progress made in MT in the last 25 years. It may be true that a lot more language pairs are covered, but usability when HQ is needed has not improved, on the contrary.

Even for main language pairs, Systran admittedly did not improve its quality significantly since 1990 or so. Small improvements are now coming from the use of statistical techniques (to try to correct mistakes in the output, or better to choose another morphosyntactic analysis in the output lattice).

Google is now proposing many language pairs built by SMT, but their reliability is even lower. Unseen words simply disappear, while they at least show in the outputs of Systran or Reverso, and can be outlined on demand. More generally, claims to the quality progress ignore the basic fact about MT, which is that **"C . A . Q << 1"** (the product of coverage by automaticity by quality is far lower than 1), which is *inherent* in the problem. To achieve HQ in large domains, one must use some kind of human help, such as interactive disambiguation in the source language, or interactive choice among a large quantity of candidates in the target language, or (co)edition of an intermediate form.

2 Inside multilingualization

The classical way to internationalize a Web site, and more generally any application, is to separate from the code the natural language strings to be shown or printed (or pronounced) by the application, and to put them in separate *resources*. One can distinguish 5 kinds of segments to localize:

- A: short, fixed texts without variables.
- B: parametrized messages with variables and possible *variants* in the surrounding text, such as error messages or personalized invites.
- C: longer and more complex texts, such as on-line help included in the code.
- D: the same, but residing not next to the code, but elsewhere, possibly on another Web site. The is the case of the minimal online help of Dokuwiki (reference manual).
- E: pages built with the tools of the considered Web site (content).

2.1 Localization of messages and GUI texts

These textual elements (A and B above) are represented as *formats*, that is strings containing variables and formatting controls, such as `"Max temperature was %f °C"` (or `"%3:1f"` to put 3 digits before the decimal point and 2 after).

2.1.1 The gettext / ngettext technique

Although it is used for Web applications such as WordPress (a CMS[1] blogging software), the `gettext` library (www.gnu.org/software/gettext/) was originally designed for standalone software. It contains a compiler of message catalogs and the `gettext` and (more powerful) `ngettext` macros. Here is an example where `ngettext` is used to generate *variants* (singular and plural forms) in the text around the (instanciated) variable.

```
// nplurals = number of variants for the plural.
// plural varies from 0 to nplurals-1,
  msgstr[plural] will be selected.
// Catalog en_US: 1 variant for plural in English
Plural-Forms: nplurals=2; plural=n != 1;
                              # catalog header
msgid "Match to %d point"     # key for case n=1
msgid_plural "Match to %d points" # --- case n≠1
msgstr[0] "Match to %d point" # n == 1
msgstr[1] "Match to %d points"# n == 0, 2, 3 ...
// Catalog jp: no variants for plural in Japanese
Plural-Forms: nplurals=1; plural= 0; // plural=0:
msgid "Match to %d point"
msgid_plural "Match to %d points"
msgstr[0] "%d POINTO MATTI"   # all n
// Catalogs for Russian : 3 variants for plural.
Plural-Forms: nplurals=3;
   # condition (OK here, false on GNU site!)
plural=(n%10==1 && n%100!=11 ? 0 : n%10>=2 &&
  n%10<=4 && (n%100<10 || n%100>=20) ? 1 : 2);
msgid "Match to %d point"
msgid_plural "Match to %d points"
msgstr[0] "..."
   # n == 1, 21, 31, 41, ..., 91, 121 …
msgstr[1] "..."
   # n == 2, 3, 4, 22, 23, 24, 32 ...
msgstr[2] "..."
   # n == 5, …, 20, 25, …, 30 …
/* Call (the same for all langages) */
printf (ngettext ("Match to %d point", "Match to
  %d points", n), n);
```

Figure 2: Example of using ngettext

In the "native" program, messages that need to be localized are tagged `(msgid, msgid_plural)`. Then, the compiler generates a file with all the messages that need to be localized. This file is duplicated and translated into every needed language. A final compilation produces a binary file used at runtime for displaying the messages in the user's preferred language. This library has been adapted to many programming languages, including PHP: WordPress is a software programmed in PHP and uses the PHP-gettext.php library.

Advantages. Because the messages are first compiled before runtime, the execution is faster than with an array loaded on demand at runtime.

Inconvenients. In the translation files, only the messages are displayed. Their context is missing. But it is often difficult to translate messages without any context.[2]

Also, in some cases, when a software is available for download, only compiled `.mo` files are included, not the source files. Thus it is very difficult to modify a translation. Furthermore, even if the source files are available, the `gettext` library must be installed locally to recompile and obtain the new .mo files.

Finally, it happens that different plugins use different localization techniques.

2.1.2 The PHP array technique

The PHP array technique is used by many CMS coded in PHP, such as the Dotclear blogging software. This technique is very similar to the `gettext` technique. Message files are in the same

[1] Content Management System

[2] That is one of the main problems encountered by IBM-Japan when localizing the giant CATIA software.

format, but each is compiled into a PHP array instead of into a .mo file.

```
$GLOBALS['__l10n']['Invalid user level'] = 'Niveau
  d\'utilisateur invalide';
$GLOBALS['__l10n']['User password missing'] = 'Pas
  de mot de passe';
$GLOBALS['__l10n']['User name missing'] = 'Pas de
  nom d\'utilisateur';
$GLOBALS['__l10n']['Invalid email address'] =
  'Adresse email invalide';
```

Figure 3: a PHP array with localized messages

Advantages. It is possible to reuse the same message in different Web pages. Also, contrary to the `gettext` technique, there is no need to install the `gettext` tools to compile the messages files.

Inconvenients. With this technique, the whole context is also missing. Furthermore, the use of an array increases the execution time of the scripts.

2.2 Inside localization of Web pages

How to improve Web sites localization? We will first compare some localization techniques.

2.2.1 The template technique

A template is the analog of a format for a whole Web page. We take as example our Jibiki platform, which is based on the Enhydra Java Web Server. Each Web page is compiled from an HTML template (`.po`) into a java class. At runtime, the `display` method of the java class is called and creates what is displayed to the user.

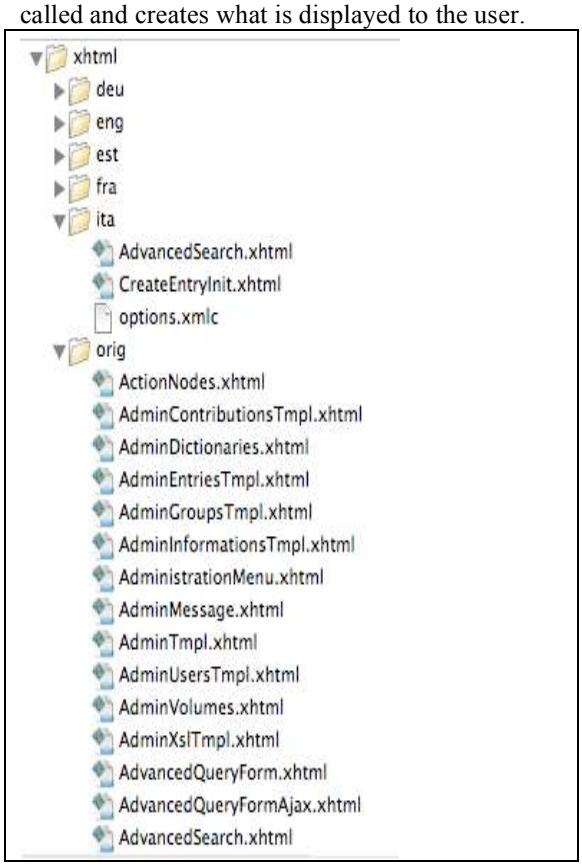

Figure 4: localization with the template technique

In order to internationalize the platform, we developed a java method that chooses at runtime which template to display, depending on the user's preferred language and the language templates available. Every page has a default template. Some templates are translated into one or several languages. The template of the preferred language is chosen first. If it is not available, the default template is used.

In theory, someone who wishes to translate a template can do so by editing it with an HTML editor, without having to know HTML. In practice, however, this method is rarely used because of several problems. (1) Many HTML editors such as DreamWeaver modify the structure of the template while editing it. It is then impossible to compile it. (2) A Web page is often built with a combination of several templates: it may be difficult to translate them, because it is not possible to see the entire Web page when translating. (3) It is not possible to recreate all the conditions in which a Web page is seen online, for example when error messages appear, because the templates are edited offline.

Advantages. Because the whole template is translated, there are no problems with the variables contained in the messages. It is also possible to translate javascript messages if their code is inside the templates.

Inconvenients. There may be inconsistencies in the translations because it is not possible to reuse already translated messages or parts: the same button with the same functionality can be translated differently depending on the page and the author.

Remaining problems. Some parts of the code are still difficult to translate, for example internal java messages targeted to the user (mainly error and feedback messages).

Also, there is no mechanism to automatically send a newly created template to the appropriate translators to translate it into all target languages.

2.2.2 Language Negociation

The choice of the language at run time can be *negociated* by the Apache (or Tomcat) server, or by geolocalization, as done by Google.

Apache Web server. The LN module serves the page matching the browser's preferred language.

Geolocalization. The results of a request depends on the user's location. If a user queries google.com from France, s/he will be redirected to the google.fr engine, etc. and the results of the same query can thus be different from one country to another.

Conclusion. Even if the LN or geolocalization choice is adequate in most cases, a multilingual Web site must always display on each page a link to its translations in the other available languages.

Furthermore, if it is technically feasible, it should memorize the choice of the user during the navigation on the site and remember it when the user comes back later.

2.2.3 Organization of resources

Another crucial aspect is the physical organization

of the textual resources in the different languages. Here is an example for the Chuwiki CMS:

Figure 5: languages resources in Chuwiki

```
; Code de la langue
Code = fr

; Noms des pages spéciales dans votre template
DefaultPage = Accueil
ListPage = Toutes les pages
ChangesPage = Changements récents
MenuPage = Menu

; Qu'est-ce qu'on affiche si la page est vide ?
NoWikiContent = Cette page ne contient aucune information

; Titre des pages
WikiTitle =  :
EditTitle = Édition :
HistoryTitle = Historique :

; Noms de boutons
Edit = Éditer cette page
History = Historique de la page
Back = Retour à la page
Preview = Prévisualiser
Save = Sauvegarder
Restore = Restaurer
```

Figure 6: resource file in Chuwiki

There are many techniques for localizing the interface elements of Web sites (called *glossary* by people working on the translation of large software products, such as FrameMaker or InDesign).
In Chuwiki, the glossary is very small, and one edits in effect a simple script that assigns values to variables. In Dokuwiki, the organization is more complex (see Figure 8). The glossary is larger, and is stored in an associative table. Elements contain variables, to be substituted on the fly. In PHP, the problem of change of order of parameters is solved if one numbers them by writing `'%s1\$'`, `'%s2\$'`, etc., thanks to an extension of the `vsprintf` function.

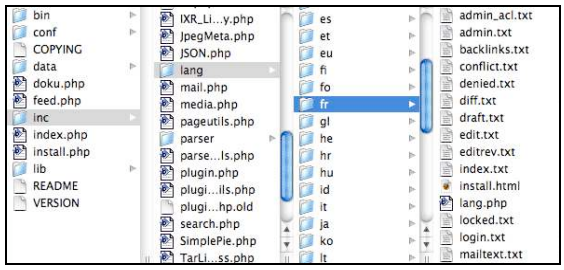

Figure 7: language resources in Dokuwiki

```
$lang['subscribe_success']     =
  'Ajout de %s à la liste d\'abonnés de %s';
$lang['subscribe_error']       = 'Erreur à
  l\'ajout de %s à la liste d\'abonnés de %s';
$lang['subscribe_noaddress']   =
  'Aucune adresse associée à votre nom
  d'utilisateur, impossible de vous abonner';
$lang['unsubscribe_success']   = 'Suppression de
  %s de la liste d\'abonnés de %s';
$lang['unsubscribe_error']     =
  'Erreur à la suppression de %s de la liste
  d\'abonnés de %s';
```

Figure 8: resource file for messages in DokuWiki

```
index.txt
====== Index ======

Voici un index de toutes les pages disponibles, triées par
[[doku>wiki:namespaces|catégorie]].
```

Figure 9: text with variables in wiki transcription (Dokuwiki)

Linking resources from Web pages through special links appearing always or on demand is a way to "turn around" the problem of context, so that one can navigate in a Web site, find the resource(s) used in the current page, and translate or correct them, knowing the context.

2.3 Localization *in context*

A more natural approach is to localize from the Web application itself. We have developed a piece of PHP code, easy to integrate in Web sites to do that. In *management mode*, a text area appears next to each localizable element. Clicking outside it lets the page appear with the updated text.

Figure 10: normal view of a Web page

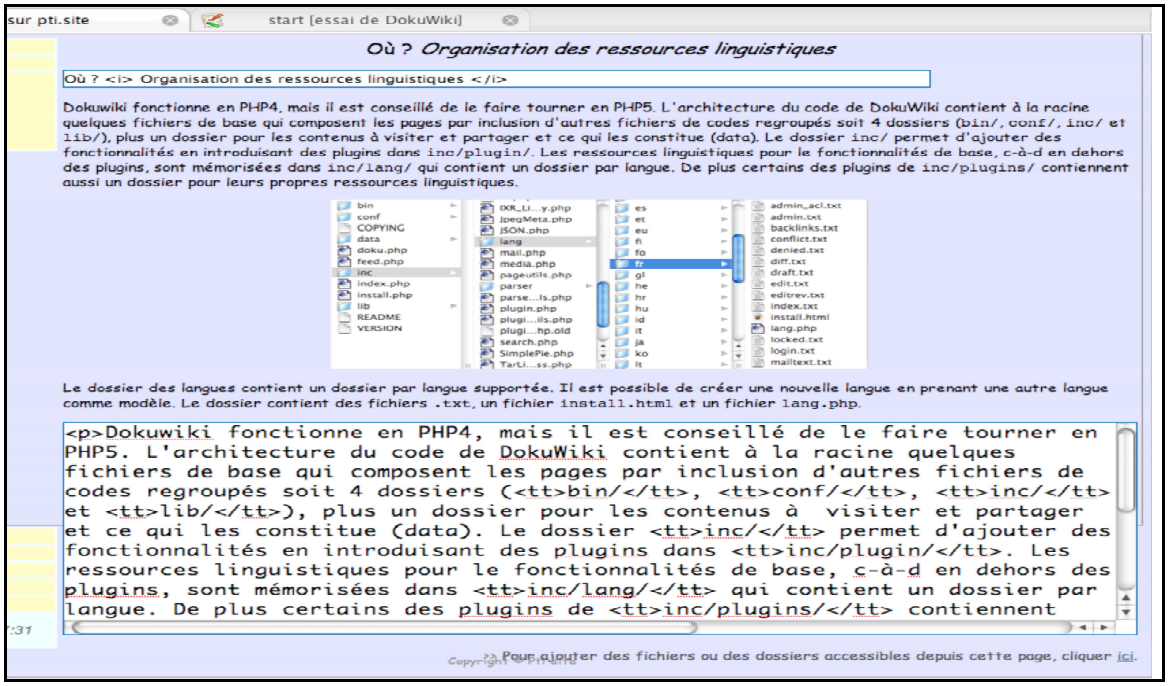

Figure 11: page in management mode, for edition

Discussion. It is frequent for a Web site to be translated by several localization techniques. For example, some CMS software use one technique for the core and other techniques for the plugins.

If someone wants to modify a translation, it is sometimes very difficult to find where it is stored: in a template, in the database, in a localization file, etc. It even may not be possible to find it when for example, the software is translated with `gettext` and the source file is not distributed.

If someone writes a translation into a new language or corrects an existing translation, there is no simple mechanism to submit the new translation to the developer of the software. Sometimes, developers provide a way to submit translation files, e.g. via a post in a forum. It would be more efficient to be able to send back automatically a new translation to the developers of the software via a standardized protocol.

None of these techniques allows a user to translate dynamically on the fly and to easily send back the translation result to the developers of the software. It is also not possible to reuse existing translated chunks of messages like with a translation memory.

3 Outside automated contributive localization & quality improvement

3.1 Call to a translation gateway

Yahoo!, Voilà, Google Translator, Systranbox and Babelfish are examples of MT *translation gateways*. The incoming Web page is segmented, segments are translated by the online MT server, links are modified so that accessed pages come back to the gateway, and a target Web page is sent on the fly to the user's browser. It is enough to copy the url of the original page into the gateway, but it is also possible for a Web site developer to add a button to some or all of the Web pages.

Advantages. The module is external so there is no need to install or compile anything.

Inconvenients. The translation engine is not tuned to a specific domain, thus the quality is usually not very good, or rather low. However, that technique is generally used only for getting an overall idea of the source page at the price of a short delay (about 0.5 to 1 second per page for Systran). Finally, there are actually not many *acceptable* available language pairs (of the 35 Systran pairs available around 2000, 8 were said to be *usable* by the EC).

3.2 Combining translation gateways

(VoTrung 2004) has produced a system to translate a possibly multilingual text, not necessarily a Web page) into a specified target language *tl*. For that, he first segments the text into parts homogeneous with respect to language and coding, and converts it into an utf-8 text with tags indicating the language and original coding of each part. Each part is then submitted to MT, after another encoding conversion if necessary.

If no MT system exists from a source language *sl* into *tl,* a double translation is attempted through an intermediate language, in general English (it may be shown to the user). The results are then assembled and a target text is produced. A Web service allows to see several translations produced by different *routes*, in the hope that errors of one system may be compensated by other systems.

One could continue along these lines, e.g. by allowing the user to correct the intermediate English form, if any, to get a better result in his language, but the quality limitations cannot be overcome.

Another approach has been developed, which concerns only the content of documents distributed through a CMS. Y. Bey has developed te BEYTrans prototype, which is a Web service for communities of volunteer translators (Bey & al. 2006, 2007) offering linguistic helps (online dynamic dictionary and translation memory, call to MT systems, online translation editor).

3.3 The iMAG concept

We propose to replace the problem of HQ certified translation by that of multilingual access, with quality increasing over time thanks to contributive post-edition.

To implement that idea, we build iMAGs (interactive Multilingual Access Gateways). An iMAG is a translation gateway, like Systranbox or Google Translator, which replaces on the fly a Web page by its *pre-translation*, and allows to browse a site in the chosen target language, by prefixing the links in the page by the url of the gateway.

Figure 12: Source and target Web page parallel display (SECTra_w)

Contrary to usual translation gateways, an iMAG is *dedicated* to an *elected Web site or domain.* That is the key to enable improvements in translation quality over time. The iMAG has a translation memory in which, ultimately, the full set of information of its elected Web site, divided into *segments*, the minimal translation units, will be stored. For each segment, we store translations at 5 estimated quality levels, with scores in each level:

- word for word (*),
- MT result (**),

and then translation or post-edition by

- a native speaker of the target language (***),
- a professional translator (****),
- a translator *certified* by the elected site (*****).

A score can be added. For example, a post-editor may want to tell that he has doubts or is sure about his work on one particular segment.

An iMAG contains a terminological and phraseological dictionary specialized to the texts of its elected site, and built from them and from the contributions of post-editors. And, last but not least, it offers an online editor, tightly integrating the linguistic helps, and offering them in a proactive way: when a segment is edited, suggestions from the dictionaries and the TM are precomputed, and MT has already been performed.

3.4 Current prototype built on SECTra_w

The scenario is as follows. Suppose a user wants to get access to page P (originally in source language *sl*) of site W in target language *tl*. S/he pastes the url of P in the iMAG dedicated to W and submits it. P is segmented into *segments* or *translation units* (TUs), which is often not trivial. Each segment is looked for in the translation memory (exact match or fuzzy match above a certain threshold). If no exact match is found, MT systems are called. If the language pair is not provided by any of the available MT, a word by word translation may be provided (*), and/or a double MT through English is produced (** with low score!), to get something rather than nothing. Having translated each segment with a known *a priori quality level*, the iMAG constructs a new HTML page by replacing each source segment with its best translation, thereby adding javascript code and other information (the source text in particular). The constructed page is sent to the user (see Figure 12). As in GoogleTranslate, added javascript code enables users to switch seamlessly

to the online translation editor, and all links are changed to allow navigation in the target language.

The difference is that translations improved by post-edition, if any, are added to the TM, and immediately available if a page containing them is consulted. Another advantage (for the future) is the possibility to build an MT system from the collected specialized data.

3.5 Towards mixed outside/inside contributive & incremental localization

Preparatory Work. There is a possibility to increase the quality of the iMAG translations by doing some preparatory work. For example, the translation memory can be bootstrapped with the existing site textual material.

Figure 13: post-edition interface in SECTra_w

Incremental Improvements. Once a first version of the Web iMAG site is settled, it is possible to increase the quality of the translations during the life of the elected Web site.

In the background, the iMAG could query some shared terminological resources in order to find High Quality translated segments and to transmit them to the elected Web Site.

Conclusion

Classical inside localization techniques are hard to apply to Web sites, in particular because developers are often not "hard" programmers. Moreover, the organization of textual resources and of the localization process varies enormously, even between instances of the same generic CMS (like Chuwiki, Dokuwiki, Xwiki, Joomla…). We have proposed and developed a technique allowing even users to localize a Web site *in context*, while using the site, in a special *management mode*.

We have also prototyped the new iMAG concept to perform contributive outside localization. Both are very promising directions for the future.

References

Bellynck V., Boitet C. & Kenwright J. 2007. *Bilingual Lexical Data Contributed by Language Teachers Via a Web Service: Quality vs. Quantity.* Proc. CICLING-2007, 19-23 Feb. 2007, UPM, A. Gelbukh, ed., 12 p.

Berment V. 2004. *Méthodes pour informatiser des langues et des groupes de langues « peu dotées ».* Thèse, UJF (thèse préparée au GETA, CLIPS), 18/5/04, 277 p.

Bey Y., Kageura K. & Boitet C. 2006. *Data Management in QRLex, an Online Aid System for Volunteer Translators.* International Journal of Computational Linguistics and Chinese Language Processing, 11/pp. 349—376.

Bey Y., Kageura K. & Boitet C. 2006. *BEYTrans: A Wiki-based Environment for Helping Online Volunteer Translators.* In "Selected and revised papers from LR4Trans-III/LREC-06 (3rd International Workshop on Language Resources for Translation Work, Research & Training)", E. Yuste, ed., Springer, 12 p.

Blanc E. 1999. *PARAX-UNL: a large scale hypertextual multilingual lexical database.* Proc. NLPRS'99: the 5th Natural Language Processing Pacific Rim Symposium, Beijing, China, Nov. 5-7, 1999, 4 p.

Blanchon H. & Boitet C. 2008. *Pour l'évaluation externe des systèmes de TA par des méthodes fondées sur la tâche.* TAL, numéro spécial sur l'évaluation, pp. 1-33.

Boitet C. 2005. *Message Automata for Messages with Variants, and Methods for their Translation.* Proc. CICLING-2005, A. Gelbukh, ed., Springer (LNCS 3406), pp. 352—371.

Boitet C. 2007. *État de l'art en traduction de l'écrit (rapport actualisé v3).* Projet TRANSAT, D-1.1.2.C, GETA, CLIPS, IMAG, jan. 2007, 68 p.

Hutchins J., Hartman W. & Hito E. 2005. *Compendium of Translation Software (directory of machine translation systems and computer-aided translation support tools.* EAMT (on behalf of IAMT), TIM/ISSCO, Geneva, 127 p. http://ourworld.compuserve.com/homepages/WJHutchins/compendium.htm

Mozilla. 2000. *French Mozilla 2000.* Official Web site for French Mozilla Project: Open Software Localization. http://frenchmozilla.online.fr (last accessed 09/05/2007)

Nguyen H.-T., Boitet C. & Sérasset G. 2007. *PIVAX, an online contributive lexical database for heterogeneous MT systems using a lexical pivot.* Proc. SNLP-07, Pattaya, 13-16/12/07, Kasetsart University, ed., 6 p.

Sénellart J., Boitet C. & Romary L. 2003. *XML Machine Translation.* Proc. MTS-IX (Machine Translation Summit), New-Orleans, 23-28 September 2003, 9 p.

W3C. 1998. *W3C Specification Translation.* http://www.w3.org/Consortium/Translation (last accessed 09/05/2007)