Creating your own Translation Memory Repository

by Ronan Martin - 25 Oct 2010

background

SAS is a major international software company specializing in data processing software. Products encompass a wide range of statistical, business intelligence and analytical products for large enterprises in all sectors. These products are localized to 21 languages. Project management, engineering and terminology-related tasks supporting the localization process are carried out in the Copenhagen and Tokyo based localization offices. The following paper describes work carried out over several years by myself, Ronan Martin, and my colleague, Krzysztof Jozefowicz, both based in the Copenhagen office.

In this introduction I would like to take the opportunity to explain my reasons for submitting a request to present this paper.

My experiences as a conference guest and discussion group participant often leave me with the feeling that in the localization domain there is a gap between theoretical and research-based knowledge on the one hand, and the practical application of solutions within enterprises on the other hand. Furthermore, many of the applied solutions that are described in periodicals and presentations are achieved by companies that seem to have invested large sums in setting up localization processes (perhaps a seldom phenomenon), or by major language providers whose life-blood depends on making major investments in the domain they owe their existence to.

My impression from talking to other conference participants is that many of them feel a sense of awe and sometimes perplexity when hearing about the available technologies being presented – but that the situation in their own organization seems to take place in a parallel universe. The will may be present to rationalize operations by introducing workflows, terminology management systems, some degree of machine translation etc., but the chaos of existing systems presents a corporate localization department with many challenges of a quite down-to-earth nature: legacy file-storage systems, ad-hoc taxonomies, hectic and sometimes unpredictable file traffic, heterogeneous technologies, ….

Wiping the slate clean and buying a comprehensive solution may be the answer for some. For others, the prospect of opening your doors to external consultants and completely re-thinking the localization workflow, and having to explain the ins and outs (and the cost) of this to an upper management that barely understands what the word "localization" means, and staff that basically likes the way things have always been done in their own particular office, must be pretty daunting.

At SAS's European Localization Center (ELC) we have developed slowly and organically over the past decade. Localization of SAS software was much less extensive in the 90's where end users tended to have a data science or IT

background and a good command of English statistical terms. In the last decade, however, end solutions have increasingly targeted the front-office business user and the need for localization has grown in step with this. To keep track of the changes and growing volume of work, ELC has expanded and it has been necessary for all of us to develop our skill set and optimize our processes on a running basis.

This presentation describes how we have tackled the seemingly mundane task of managing our translation memories, and the unexpected benefits that have arisen from this initiative. The path that we have taken does not cover new theoretical ground and is not earth-shattering in any sense. But I would like to open a window onto a solution that has evolved in the day-to-day maelstrom of the localization workspace and hope that it can inspire others to find a low-cost homegrown solution that knits into the idiosyncrasies of their particular localization workflow.

TMs: the situation before….

SAS uses three different CAT tools to localize software products. Around 2004 we used a great deal of time and effort in re-organizing our storage structure for projects being sent out for localization, and translated projects coming back. The major part of this effort involved lengthy and detailed discussions about object granularity (product domain, product, project, toolkit, filetype etc.) and a suitable taxonomy for these objects. I mention this somewhat basic step because in my experience this is the absolutely most important make-or-break factor in creating an automated workflow of any kind. You can muddle by on inconsistent naming systems, but in the long-term this will continually trip you up when you try and introduce overarching systems that are scripted in a programming language.

We created a good structure which was partnered with a database using the same set of taxonomies. Using the database and Perl scripts and a Web front-end we set up a fairly straightforward automated workflow. Our translators were able to trigger the building of localization toolkits, download these, translate them and upload them again. [A localization toolkit is a package containing all the files needed to deploy a localization project in a CAT tool on a local computer. The returned, completed toolkit is the same package including the translated files and the translation memory.] The system ensured that the uploaded toolkits and the files they contained were placed in the right folders. Some of the files were the translation memories. These were placed in designated folders but left in their proprietary format.

After some time we wanted to set up a terminology management system, and needed to carry out term extraction on our existing translation memories. It now became a problem that our translation memories were distributed in the system and stored with each project. But pulling them together didn’t fully solve the issue because they were still in a binary format that was difficult to work with.

We decided to go the TMX way. We added an extra step to the toolkit upload process which took the uploaded project and used API's from our CAT tools to create TMX. This is in itself was not an easy format to play around with, although now the TMs from the different tools were at least uniform. We decided that we needed a simple delimited txt format that was more amenable to scripting and we created an additional XSLT transformation that converted the TMX files into a very simple delimited format with the source and target strings only.

Other metadata was stored in the file name (version, project code, filetype), and additional metadata was also obtainable through the relationships in our data structure (domain, project ship date etc.)

Whenever a project was uploaded by a translator, a TMX file was created for that project/language/version and placed in a designated folder. We allowed uploaded files to overwrite any existing files for the same project/language/version, as we only wanted to store the most recently uploaded TM in these files, but wanted to allow different version numbers to endure.

We were now close to a stage where we could aggregate all the TMs, but first needed to go back and trigger the TMX/txt process for all legacy projects as the only trigger we had until now was a project upload and many projects are dormant between releases. This was actually quite a laborious task involving hundreds of projects.

Now we created a script that ran nightly. It combed the project folders and identified and gathered together all the txt format TMs for all the latest versions of projects. This made it possible to aggregate the files into a grand data set for each language. The rows of these data sets consisted of the source string, target string, version, ship date, project name, filetype, toolkit name, product domain. At SAS the development language is English and therefore the source language for localization is always English. In other words we had a set of language-specific TM silos containing the latest version of the TM for every product ever localized at SAS, in an easily readable format.

An interesting feature of our automated workflow is that our translators can build toolkits and upload translated toolkits themselves. Within the scheduled period, until the sign-off date, translators can upload toolkits as many times as they wish. This ensures that the TM stored in the TM repository can be kept updated by translators to within a 24-hr period.

Although the original reason for creating these TM repositories was to extract terminology, we quickly became aware of the possibilities embodied in these data sets which were updated nightly.

now that we have TM repositories, what else can we use them for…?

Like all localization centers, we receive many queries from translators, testers, reviewers, etc. about source/target strings. We added an extra step to the repository creation and also created large language-specific txt files (one for

each language) containing all the strings from the TMs. This made it very easy to quickly locate the offending string pairs and answer many kinds of query.

Interest in accessing these files snowballed, and we realized that it would be of major benefit to our translators and others to make the files available in some way via Web access.

The solution we came up with was to create a Web form that used a CGI script to send queries on to a Perl script. The Web page provides a set of search fields. Users enter search strings and these are sent for processing to a script which uses regular-expression pattern matching to find results. These are returned in a new Web page. We added a number of subsetting possibilities. Users could select a language, a domain or project or just specify All, and search in source and/or target strings. We also added various kinds of advanced search.

This tool, which we call TM Search, was very heavily accessed and we were surprised at the wide range of users. Word spread within the company and we found that many other groups outside the localization community were also using it.

We have also made the tool available to our language service providers and it has drastically reduced the volume of terminology queries previously sent to us. We also feel that it is contributing greatly to increased consistency in the translations. As well as having the chance to see how other translators within one's own language have translated certain strings or terms, translators can also search in languages that are related to their own and see how the translations were tackled there.

Customers have also become aware that this facility exists and have expressed interest in having access to it. Many SAS customers operate across national-language boundaries, and local offices in, say, Holland, Germany and France often sit using the same software product, carrying out the same process. But the product versions they use are localized to their own language. Having look-up access to the English source for the localized UI text strings they see on their screens would greatly assist cross-border communication.

addressing an old pain: which TMs should I use?

There was a constantly recurring issue in the localization process. SAS products are released in waves that we call ship events. For each new ship event we typically target between 10-20 products for localization to a number of languages. Products are only localized to a certain language if there is a good business case for doing so, so the constellation of languages changes from project to project.

We had always tried to advise localizers about which existing translation memories would reduce the word count for the new localization projects we launched. A previous release of the same product was an obvious candidate, and if we knew of related products these were also recommended. However, quite often, half-way through the translation cycle, a translator would contact

us and tell us that they had discovered that attaching the translation memory from project XXX reduced the translatable word count by, for example, 6000 words.

In short, we had an idea about which TMs may be of help, but this was based on what various people in the organization might remember, or vague communications filtering through to us that developers had re-used several components from product A, when creating product B. For new product releases we found it hard to recommend TMs to attach as reference material. This situation was compounded by the fact that for each language group we had a different set of TMs.

We realized that the TM repository may provide us with a way to systematically investigate the nature of our TMs.

Initially we toyed with the idea of a solution that created a purpose-built TM: purpose-built for each new project in hand. This would be done on the basis of strings from the repository. The format of the purpose-built TM could, for example, be TMX; though right now the repositories were stored in a database format, or txt delimited files.

We looked into the possibility of going back to the TMX and aggregating this on a native XML server. With this in mind we invited an XML software provider to advise us on a solution. The architecture could be set up. The solution was sophisticated but somewhat expensive, and would require that we learn XML technologies like XQuery, XPath, etc. to an expert level, or commit ourselves to purchasing ongoing consultancy services from the provider which would bump the price up even higher.

We took a look at the tools and technologies that we already had at our disposal. We had the SAS environment and the SAS programming language, and the necessary programming skills to attempt a solution. The solution we came up with turned out to be a fairly simple one.

We decided to abandon the approach of creating purpose-built TMs for projects based on all the existing TM material. Because of the nature of our products, preserving the context of a translation is very important. We were concerned that if we allowed an automated process to pluck out strings individually from the repository, the resulting TM would encompass too many different contexts from different products and the result of pre-translations would be internally inconsistent. Also, this process would result in very many variant target strings, and our translators would have to sit and choose between these – a time-consuming task.

Another consideration was technical in nature. Attaching a proprietary translation memory to a tool is straightforward. The proprietary TMs tend to be richer in data (some of this is invariably lost or modified when changing the TM to another format). Wherever possible we prefer to attach TMs written in the tool's own format.

What we were looking for was a solution that emulated the existing process. We wanted a process that would return results in the form of a short list of

TMs that would provide the greatest possible number of pre-translations. We could then use the proprietary TM formats and attach these to the new translations – a very quick and easy step.

TM Discovery

We called the new process TM Discovery.

It consists of a number of steps that are scripted in the SAS programming language, which is perfectly suited to crunching large data volumes, but could just also have been written in many other languages. The basic algorithm is quite simple.

Our localizations are packaged in toolkits. Each toolkit contains files that are the same file type. A typical file type could be a Java properties file. So, for the new release of product AAA, we wanted to discover which existing TMs would give us the greatest number of pre-translations for our AAA_properties toolkit for, say, French. This is the same as asking: are there any comparison matches between source strings in AAA_properties and English source strings in the French TM repository.

The repository contains all the string pairs for all translations that have been carried out to date in French. The rows have a "product" variable and a "filetype" variable, but also a "toolkit_name" variable which is a combination of project and filetype. Imagine a string in the BBB product taken from the properties files: the toolkit_name would be BBB_properties. This is an important point because for us the toolkit, designated by the toolkit_name, represents the basic-level granularity. Below it we have individual files and eventually individual strings, and above it we have product, domain. This is important in deciding how to deal with duplicate strings, which we did as follows.

If we look at source/target string pairs (as opposed to source strings only), a duplicate row was defined as a row in which the source string and target string were identical with the source string and target string from another row with the same toolkit_name. In other words all the string pairs at the BBB_properties level were unique. If two rows contained the same source string, but a differing target string these were not considered duplicates.

Coming back to our new set of strings under investigation, AAA_properties, the first step is to copy the repository data set for French to a playpen and append the set of AAA_properties strings under investigation as new rows. Now a sort operation can be carried out on the source string variable. Duplicate source strings will be grouped. If a group contains a string from AAA_properties, then it is of interest to us. We keep these groups as SUBSET_1 and remove all other rows.

Now we carry out a sort operation on the new SUBSET_1, but this time on the toolkit_name variable so that the resulting groups of strings are each a set of strings belonging to one toolkit. These groups are counted internally, and the largest group (excluding of course AAA_properties) is extracted and we can

call this SUBSET_2. For example, this could be a set of strings from the XXX product, called XXX_properties. SUBSET_2 is reduced to only include unique source strings. This is the top candidate TM and can be added to the top of a list of recommended TMs. A metric can be added to this: number of segments matched, number of words matched.

Now we return to SUBSET_1 and using SUBSET_2, mark off all rows in SUBSET_1 that match a string in SUBSET_2, and remove these rows from SUBSET_1. The reason for this is that we are giving a privileged status to strings from XXX_properties (found in SUBSET_2). It appears that XXX_properties is the most similar TM to the set of strings we are investigating. We do not want to encounter these strings again as we continue our investigation through other TMs. They are, so to speak, already spoken for.

Now we perform the same operation as before on the remainder of the strings and create a SUBSET_3. SUBSET_3 turns out to contain strings from YYY_xml and this is written to the list as the next-best candidate TM, and the metrics are worked out for the number of matches. There is thus no "measured" overlap between XXX_properties and YYY_xml. There may in fact be overlap, but this is not reflected in our metrics. We are interested in the total number of words that can be pre-translated using the smallest possible selection of TMs.

Typically we find that after the first 3-4 recommended TMs, the number of matches falls to less than 50 words, and then very quickly to less than 5 words, petering out after 6-10 recommended TMs.

The list of recommended TMs is then saved as a txt file and stored in a folder below the project information Web pages for the AAA project. A link to this file is added to the actual information pages. The operation is then carried out for other filetypes that are being localized to French for the AAA product. Typically we localize 3-8 filetypes for each product. Then the whole operation is performed for each other language into which the AAA product is being localized. All these operations are prepared and executed as a batch job.

So, localizers and PMs open the project information Web pages for the AAA project, and under a designated tab are presented with a list of links. They are arranged by language, and within each language by filetype. Clicking on a link opens the list of recommended TMs that should be attached as reference material to a new project.

discussion of results

Typically, the TM Discovery returns a list of 3-4 useful TMs. The top candidate will most often be the previous version of the same toolkit. So for the AAA_properties_2.0 toolkit, the top recommended TM will be AAA_properties_1.0 – in other words the previous version of the same toolkit. This is a re-assuring result, as any other result would invalidate the process.

But often the secondary recommended TM contains substantial matches, and this would not have been predictable. For new products and sometimes even for new releases of existing localized products we discover major overlaps that we had not been aware of between the current project and an already translated product.

Results are different for each language – sometimes quite different. As described above, this is due to the fact that the set of existing TMs varies from language to language. Previously we could give some indication about which TMs may be of benefit, but this was based on our knowledge of the English source, and didn't always make sense for some languages. We are now able to provide a language-specific list of recommended TMs.

The metrics provided by the TM Discovery process can be taken as a guide to the usefulness of the TM. The inter-relationship between the TMs on the list returned is a solid and reliable one. However, the metrics will never match completely the metrics obtained from the CAT tool once the project is created and the reference TMs leveraged. This is because the TM Discovery tool does not attempt to carry out fuzzy matching and there can be discrepancies caused by different handling of things like case-sensitivity, punctuation, etc., between the tool and the TM Discovery process.

Results returned can indicate that TM in one tool-specific format should be used in another tool. This necessitates us using the TMX version of the TM, which for some tools requires an intermediate conversion, but this is no great problem.

the cost

We did not purchase any additional software or hardware. Naturally there is the time spent creating the various scripts, but this was done in stages over a fairly long period and did not create any noticeable resource problem.

We have not carried out any in-depth analysis of the savings gained by having the system. The TM Search tool saves valuable time for many different types of user and has increased the internal consistency of localizations. Quicker troubleshooting means more bugs can be addressed in a release and this in turn means better quality.

The TM Discovery tool displays a more immediate effect. In the word count reports obtained after attaching recommended TMs, it is possible to see how many words are pre-translated; words which may otherwise have been missed. At a cost of 5-6 words per Euro, for just one localization project, if an unknown TM relationship is discovered to the tune of 3000 words, if the project is earmarked for localization into 12 languages this means a saving of around 6,000 Euros. At ELC we manage around 250-300 localization projects annually.

Krzysztof Jozefowicz is the Localization Manager at ELC, SAS's European Localization Center. He has worked in the localization industry for 15 years and has an M.Sc. in Computer Science from the Jagiellonian University in Cracow.

Ronan Martin is the Terminology Manager at ELC, SAS's European Localization Center. He has worked with terminology and localization for the last 10 years, and previously as a language-learning consultant and translator. He has an M.A. in Educational Psychology (Pædagogik) from the University of Copenhagen.