

Case Study: Translation of a Database Manual using MTM

Situation

Documents to be translated – From our client, we received a number of XML files containing more than 300,000 words to be translated from English to German. These files comprised the user manual for a database system.

Working environment – Our client uses Idiom Worldserver as their translation environment (translation memory and workflow). Based on our experience, the language combination, and the type of text, we choose Lucy Software’s machine translation engine, which is a rule-based system.

Since the client is running Idiom Worldserver, we had no access to the API. In other words, we could not use Worldserver functions to call the MT engine.

Considerations

Ease of use – The addition of an MT engine to the working environment is supposed to make the life of the translator easier, not harder in any kind of fashion. The translator should just get additional translation proposals when he/she needs them and not be confused by additional, poor matches when good matches exist. In no case, existing translations (100 % matches) must be obscured by the MT output.

Memory pollution – With every translation, the translation memory grows. Thus care needs to be taken not to increase the size of the memory by useless (or never used) matches.

Complications

Function words – The text contains a lot of function words, e.g. commands, that must not be changed or translated at all. Since they happen to be appearing in the middle of a sentence quite often, we could not just “hide” them from the MT engine (which would ruin the analysis of the source sentence), but had to mark them as “constants”. These “constants” are treated by the MT engine as nouns that are similar to proper names.

Program code snippets and index entries – On top of the function words, the text contained quite a few program code examples, which we needed to hide from the processing by the MT engine, and index entries that had to be separated from each other in order to enable the MT engine to translate them properly.

Solution

Standard approach: Pretranslate and import into memory – The direct (and usual) approach is to send all the files through a batch translation by the MT system first. Afterwards, you align the source files and the MT translations and import the bilingual files into the memory. Apart from alignment issues, this has two major drawbacks:

- ⬇ The memory size is increased and the memory includes a lot of unwanted “matches” that keep popping up confusing the translators.
- ⬇ Translators will be provided with MT “matches” even if 100 % matches exist.

Statistic or rule-based MT?

Independent research shows that both MT systems deliver comparable results that tend to be insufficient for direct use and hence should be used in connection with TM and postediting. There is no general recommendation on what type of MT system to use, but there are selection criteria:

- 1) Do you have huge corpora of bilingual texts (TM) covering one subject matter area?
- 2) What language pair(s) do you need?
- 3) What text types are you translating?
- 4) Who codes the terminology?

In this case, the language pair English to German and the texts including many function words led to the decision to use a rule-based MT system.

Batch mode or on the fly?

Machine translation can be performed in batch mode upfront. In other words, the whole text is pre-translated by the MT engine and then provided for post-editing. The MT engine can also be connected to the TM system requesting a translation of each segment from the MT engine on demand.



Automation: The automation steps described here (and quite a few others) can be achieved by using Lucy's Pattern Matcher, a clever search & replace tool or with an editor that provides regular expression searches (MS Word, Notepad++, to name just 2 of them). They need to be created just once for each text type and require surprisingly little creation efforts.

Terminology coding used to be the predominant cost factor. In case of Lucy Software, this pain is considerably relieved by defaulting: the engine quite successfully defaults grammatical information. For this project, we had to code more than 4,000 entries, which would have been impossible without defaulting

Savings: The savings (S) achieved can be calculated as follows:
 $S = HC - TC - AU - AM$
where:
HC = Time saved by post-editing instead of translating multiplied by the hourly rate
TC = cost of terminology coding
AU = cost of automation coding
AM = amortization of MT engine

Eule Lokalisierung GmbH
Holstenstrasse 104
24103 Kiel
+49-431-99042-0
www.eule2005.de
info@eule2005.de



Our approach: Protect existing translations – Before we sent the files to the MT engine, we prepared them carefully in order to protect existing translations and to prevent function words and program code from being translated.

In detail:

- Step a)** Export the source files with existing translations as bilingual files, i.e. files containing pairs of English and German sentences.
Remark: The "German" sentences, in reality, are German only if a translation already exists. Otherwise, they are just a copy of the English source sentence.
- Step b)** Protect existing translations
- Step c)** Mark function words as "constants", i.e. words that are not to be translated, but treated as "constant nouns" by the MT system.
- Step d)** Hide all program code from the MT system.
- Step e)** Isolate individual index entries into separate segments

All these steps are automated and take little time and efforts.

Then we had the MT system translate the prepared files. Actually, the MT engine only translated the "German" sentences that were not German at all (see remark above).

On completion of the machine translation, which took just a few seconds, we removed our protection marks (again automatically) and re-imported the pretranslated files into Idiom Worldserver.

What did we achieve?

- ✓ We had not added anything to the memory before the actual (human) translation/post-editing step. Thus the memory includes verified segment translations only. There was NO memory pollution at all.
- ✓ We had no stupid "translations" of function words. Hence a command like SELECT * FROM TABLE was left as it was and not translated into something like WÄHLE * VOM TISCH, which would be completely meaningless to the database.
- ✓ And we had the perfect environment for the translator: 100 % matches were still 100 % matches (marked in blue), fuzzy matches from the TM were still presented as usual (marked in brown) and MT translations were already inserted and marked as "manual translations" (green). In this way, it took our translators just one glance to decide whether to leave the translation as is, check out the fuzzy matches, or edit the so-called manual translation, in reality the machine translation.
- ✓ **SAVINGS:** We were able to offer a 20 % discount on all MT-pretranslated segments to our client. In other words, the client never had to pay the full price for "new words".