Integrating Machine Translation with Translation Memory: A Practical Approach

Panagiotis Kanavos
Freelance Technical Translator
K. Varnali 20, K. Aharnai, 136 71
Athens, Greece
panoskanavos@gmail.com

Dimitrios Kartsaklis
Freelance Software Engineer
M. Mpotsari 46, Petroupolis, 132 31
Athens, Greece
dimkart@gmail.com

Abstract

The purpose of this work is to show how machine translation can be integrated into professional translation environments using two possible workflows. In the first workflow we demonstrate the real-time, sentence-by-sentence use of both rule-based and statistical machine translation systems with translation memory programs. In the second workflow we present a way of applying machine translation to full translation projects beforehand. We also compare and discuss the efficiency of statistical and rule-based machine translation systems, and propose some ideas about how these systems could be combined with translation memory technologies into a unified translation application.

1 Introduction

Machine Translation (MT) has been evolving rapidly, drawing attention from other professionals besides researchers. However, we cannot claim that it is generally accepted as a useful and productive technology by the professional translation industry. On the one hand this is partly due to unrealistic expectations as to what MT should be able to do and how it should be able to achieve it. On the other hand, it is true that the MT community focus mainly on research, overlooking practical issues. This is reflected in various online surveys and forums related to professional translation, where translators and language service providers consider MT an experimental technology suitable for draft translations of e-mail messages and web pages. To this respect, Computer-Assisted Translation (CAT) vendors have been reluctant or uninterested in investing further in MT technologies, with the exception of Google's MT service which is offered as a plugin to a few CAT programs.

We will demonstrate that MT–when combined properly with Translation Memory (TM) technologies–is actually a very useful and productive tool for professional translation work, contrary to the common belief that MT is still restricted to draft Web translations of low quality. Furthermore, we will show many possible ways to combine MT systems with TM workflows, as well as the ability to use both rule-based and statistical MT systems.

2 Methodology

Our workflows and methodologies are divided into two general categories: the on-demand, sentence-by-sentence application of MT, and the one-time application of MT into the whole translation project. In both cases, we use and combine various tools, MT systems, even operating systems and platforms, to provide a unified translation environment which most professional translators are familiar with. The CAT programs used are Swordfish II (a Java application) in a Linux operating system, Déjà Vu X in a Microsoft Windows environment, and Wordfast, a Microsoft Word macro template, also used in a Windows operating system. The MT systems used are the Moses statistical system (Koehn et al., 2007), and a rule-based edition of Systran[1].

[1] Although the Systran edition that was available for our language pair was rule-based, newer editions use a hybrid technology incorporating both statistical and rule-based elements.

Ventsislav Zhechev (ed.): *Proceedings of the Second Joint EM+/CNGL Workshop "Bringing MT to the User: Research on Integrating MT in the Translation Industry" (JEC '10)*, pp. 11–20. Denver, CO, 4 November 2010.

For the purposes of our research, we chose to translate two technical books in the field of informatics from English to Greek. The first one was an instructive guide (referred to in this paper as *Book 1*) to a popular desktop application, which contained a lot of steps for accomplishing specific tasks and had a certain amount of repetitive or similar text. This kind of text is ideal for work in a TM environment since the TM yields a lot of exact and fuzzy matches. The other book was an academic one (referred as *Book 2*), covering more complex and theoretical topics in the field of Computer Science, and contained very few repetitive sections or step-by-step instructions. Such texts do not benefit particularly from TM matches, although they do benefit from other features offered by CAT tools, such as concordance searches, consistency in style and terminology, automatic number and tag checks, etc. The configuration of the CAT programs was almost identical: we used the same TM which contained 140,000 translation units, the same terminology database containing an average of 30,000 entries, and the same segmentation rules. All our resources derived from previous translations in the Informatics domain. Also, the TM fuzzy threshold was set to 70% for all programs.

We translated both books in our three TM-MT combinations and obtained measurable results in terms of time spent, translation quality, and overall efficiency of our workflows. It should be noted that the books translated with our workflows were real assignments from a publishing house and they are now under publication. We divided each book in six equal parts (two for each combination) and we created separate translation projects for each TM-MT combination. Then, in each combination, we translated one part using only the TM and another part using both the TM and the MT, keeping track of the time spent so we could measure the effectiveness of the MT. Productivity is expressed in target words per hour.

2.1 Applying MT on demand during the translation process

The ability to use MT engines from within CAT programs is largely dependent on the CAT program itself. Thus, there has to be a way to connect the MT engine to the TM application and invoke it at will for selected segments. For this reason, we chose the Swordfish II CAT program, which offers a flexible plugin architecture allowing the user to connect to external applications or scripts, and Wordfast Classic, which provides a feature for connecting to MT systems. Although both programs are used under the same method, their combination with MT systems varies significantly because of the operating systems involved and the MT systems they can be connected to. Swordfish was used with Moses, while Wordfast was used with the Systran engine. These particular combinations were chosen based on the tools we had available and some practical factors (explained in the following sections) concerning the integration of each CAT tool with the corresponding MT system.

2.1.1 Swordfish's translation environment

Swordfish's translation environment offers all the usual features met in CAT tools, such as the translation grid where the segmented text is presented for translation, the TM database results pane, the terminology database results pane, and an Auto-Translation pane, which shows the results of the program's attempt to assemble translations using matches from the TM and the terminology databases. This is of special interest for the purposes of our work since it is the closest native feature the application offers to MT. Another feature we will use for comparison is the application's GTranslate plugin which can be invoked by the user on-demand for automatic translation using Google's MT engine.

The workflow presented by Swordfish is typical of CAT programs:

- Import, conversion, and segmentation of the source files into Swordfish's native format (XLIFF)
- Connection of TM and terminology databases into the project
- Pretranslation of the entire project using the connected TMs (optional)
- Translation of the project, segment by segment
- Export the project into the original file format

Swordfish integrates seamlessly with any external application or script due to its plugin architecture.

This is a key feature that let us customize our workflow to a great extent. We created a custom Python plugin that manipulated the current segment. The exchange file created by Swordfish is in the XLIFF format (XML-based), so the translatable text can be easily extracted using Python's XML modules.

The XLIFF project format used by Swordfish is very flexible for applying MT since it does not require any special conversions or additional preparation steps. Furthermore, integrating the MT results into Swordfish's environment was seamless, since we could automatically create an appropriate XML element for each machine-translated segment and make the translation appear into the TM matches pane.

2.1.2 The Moses Statistical MT System

Moses is a statistical MT system developed by the EuroMatrix[2] and EuroMatrixPlus[3] projects and various institutions and universities and is licensed under the LGPL (Lesser General Public License). Such systems require large corpora of text for training and creating phrase tables and language models which are used during the automatic translation (decoding). Translators can benefit from such systems since they already have available considerable amounts of texts in the form of TMs. All CAT programs offer an option for extracting the entries of the TMs into the industry-standard TMX format which can be easily manipulated for preparing the text corpus.

Moses, along with the required utilities and scripts, requires a Unix environment and a fairly capable computing system. These requirements can impose practical restrictions for wider adoption of this workflow, since most translators use PCs and laptops with limited computing power while few of them work in a Unix environment. This issue is also discussed in Section 4.1.2. We installed and configured Moses on a Linux Server machine with a dual-core processor and 8 GB RAM, in a LAN environment.

The steps we followed for preparing an MT system with Moses are outlined in Figure 1. The parallel corpora extracted from our TM contained about 140,000 sentences, an amount not particularly large for statistical MT training. However, the efficiency of our MT system was not hindered by the limitation in the amount of sentences because the training corpus was very domain-specific to our translation project. This issue is extremely important when combining MT and TM technologies and it is analyzed in greater depth in Section 4.

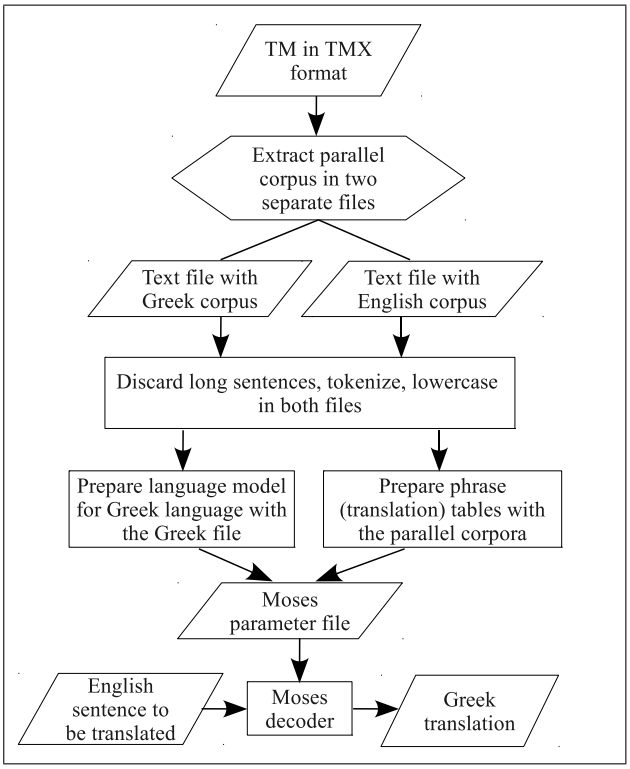

Figure 1. Steps for preparing a Moses MT system

2.1.3 Integrating Moses into Swordfish

The connection of the two systems was achieved with the custom Python plugin that extracted the translatable text from the temporary XLIFF file generated by Swordfish for the current segment and sent it to Moses for decoding using the XMLRPC API. The translated text was then copied to the XLIFF file which, in turn, was inserted back to our current segment in Swordfish's translation grid.

Our workflow with this configuration can be summarized as follows:

- We translated our project segment by segment (pretranslation with the TM was not applied)
- TM matches above 80% were always accepted and the proposed translations were edited as necessary

[2] See `www.euromatrix.net`
[3] See `www.euromatrixplus.eu`

- When the TM offered matches below 80%, the quality of the match was evaluated and the proposed translation was either accepted and edited or rejected
- When a TM match was rejected, MT was applied to the segment and the result was edited accordingly
- For segments with no TM matches, we always applied MT and the result was either accepted and edited or rejected (in that case we had to translate ourselves the whole segment)

Our decision point of 80% for the fuzzy threshold was selected based on our previous experience with similar setups. Up to this fuzzy threshold, Moses proposed translation is usually quite accurate and comparable with the TM match. This happens because the TM used in the current project–and thus the matching translation unit–was also used for training our MT system. Furthermore, the translation unit is also contained in our MT's language model. Above this fuzzy threshold, the TM match is usually very close to the desired translation, so editing the proposed translation is more time-efficient. The 80% threshold was also suggested in a similar study by Carl and Hansen (1999).

A special pattern was followed for testing every feature our environment could use to provide a suggested translation. In particular, we randomly selected 50 fuzzy matches with a similarity percentage equal or higher than 80% and source sentence text length between 25 and 30 words and compared the TM match, Moses proposed translation, Google's MT (utilized with Swordfish's GTranslate plugin), and the application's "Auto-translate" feature which combines matches from the databases and assembles translations. The results we obtained from this sampling are listed in Figure 2.

The time spent for completing the translation followed a trend typical of our previous work with such technical books in TM workflows, and varied between the two books. In particular, the translation of the instructional book (Book 1) required less time than the translation of the academic one (Book 2). As it is illustrated in Figure 3, our productivity was higher in the instructional book translation due to the less complex subject and the higher amount of

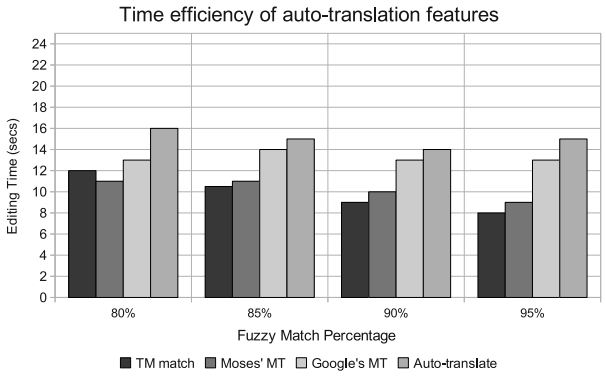

Figure 2. Comparison of all available auto-translation tools and features

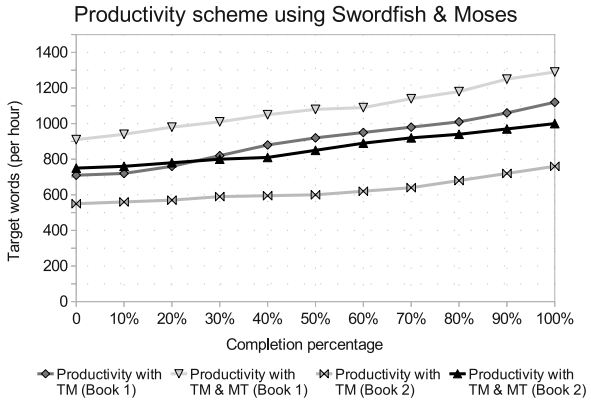

Figure 3. Productivity scheme with the Swordfish and Moses combination

repetitions which resulted in more TM matches as we were progressing in our work. Furthermore, our productivity starting point was higher in the instructional book (700 target words per hour) compared to the academic book (550 target words per hour).

2.1.4 Wordfast translation environment

Wordfast Classic is a Microsoft Word macro template that works within MS Word. Thus, the documents to be translated cannot be imported in a project but rather they are directly opened in MS Word and translated one by one. Despite the fact that Wordfast Classic is restricted to the file formats recognized by MS Word, it offers a full-featured translation environment with TM, terminology integration, and real-time quality checks. Furthermore, it allows a great deal of customization via macros which can be invoked at various stages of the trans-

lation process and provides integration with MT systems.

2.1.5 The Systran MT system

Systran is a widely known MT application that is available in various editions and language pairs. The latest editions of Systran include a much improved hybrid MT engine with many customization options, and the ability to create custom user dictionaries. The version we used for this work, however, was using only rule-based MT technology. The entries in the user dictionaries are not merely used as word-to-word translations, but rather they are subject to the MT engine's rules and thus they can generate inflected forms. This provides a great means for improving the MT engine but also imposes a restriction: building the user dictionary before actually working on a specific translation project is rather tedious and time-inefficient for the following reasons:

- The translator can't possibly know how a particular term or phrase will be translated in a specific context by the MT engine
- The custom dictionary entries should be tested in a sample sentence
- Inflected forms of some entries cannot be translated correctly by the MT engine so other means of improving the engine should be applied (for example, post-editing macros–see the following section)

It should be noted, however, that after building a custom user dictionary for a specific domain it can be used and tweaked for future projects in the same domain.

2.1.6 Integrating Systran with Wordfast

The integration of the two systems was achieved via a special Wordfast's feature that works as follows: if there is no TM exact or fuzzy match for the segment currently translated, then the segment is transferred to the MT engine and the translation is copied into the target segment.

Our system was also improved by using two sets of custom Word macros. The first set included macros that accomplished various pre-translation tasks which were executed to the source segment before submission to the MT engine. Such tasks involved the temporary substitution of source words or phrases that couldn't be translated properly by Systran's engine with other similar that didn't alter the sentence's meaning. For example, the engine could not translate correctly the auxiliary verb "might", even though the word was properly coded in our user dictionary. Our macro was simply substituting "might" with "may", and after the translation was returned to our segment the original word was restored. The second set of macros executed post-translation actions to the translated segment returned by Systran. These actions improved the style of the translated text by various ways, very specific to our target language.

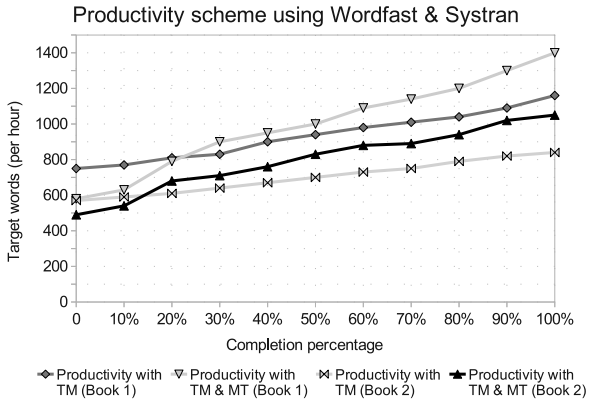

Figure 4. Productivity chart using the Systran and Wordfast combination

Our productivity is presented in Figure 4. As it is illustrated in the chart, our productivity was lower at the beginning of our work. This was due to the time we had to spend to build the custom Systran dictionary. When we had completed about 20% of the translation and had added 80 entries in our custom dictionary, our productivity increased to our usual standards. Beyond this point our productivity increased drastically while the number of entries we had to put in our dictionary was constantly decreasing. By the end of the translation our dictionary contained only 240 entries.

2.2 Applying MT to the full project

The implementation of this approach does not necessarily require a plugin or some other means of connecting the CAT program to MT systems, although

such a feature would be more convenient and restrict the number of steps required to prepare the project file. The number of steps is also greatly reduced if the application's project file has a standard or open format such as XLIFF or other XML-based format. For the implementation of this approach we used Déjà Vu X, a Microsoft Windows application based on the Jet Database engine.

2.2.1 Déjà Vu X's translation environment

Déjà Vu X is based on a concept similar to Swordfish's. It offers a two-column translation grid with the source segments in the first column and empty cells in the second column where the user can type their translations or insert matches from the TM, as well as TM and terminology match panes. The application also includes a special feature called "Assemble" which applies EBMT (Example-Based Machine Translation) algorithms for converting fuzzy matches to exact by combining TM and terminology entries.

2.2.2 Using Moses with Déjà Vu X

Déjà Vu X does not offer a way to connect to external applications or scripts, so using MT segment by segment is not possible. Instead, the only efficient way to use Moses with this CAT program was to extract the whole project file and pretranslate it with Moses. The application offers an option to export the whole project file into a table in RTF format. Each table row consists of three fields and corresponds to a segment. The first field (cell) is a unique row ID number which must not be altered in any way because then the application cannot re-import the file back to the project. The second field is the source segment, which includes the source text and the internal formatting tags, and must not be altered too. The third field is the target segment where a translation can be typed in. Before we exported the project, however, we applied pretranslation against our translation memory with a fuzzy threshold equal to 80% and then we locked the pretranslated segments to avoid exporting them with the rest of the project. With this approach the number of segments was reduced to some extent so Moses had to translate fewer sentences. The workflow is outlined in Figure 5.

An alternative workflow that may seem more reasonable at first glance would be to export all the segments without applying pretranslation; translate the whole project with Moses; import the translation back to the project file; and then compare Mose's translations with the TM matches and keep the best translations. However, the benefits with this approach would be minimal because above the 80% fuzzy threshold it is more time-efficient to accept the TM matches and edit them as necessary.

When the file was translated, we reversed the order of the preparation steps in order to re-create the RTF table and place the machine-translated segments to their corresponding cells. Then we imported the file back to our project and started working segment by segment. It should be noted that the fuzzy match threshold was lowered to our standard 70%. This resulted in some segments which were filled with MT translations and also had fuzzy matches between 70% and 79%. Now our workflow was quite different than the one followed in the Swordfish-Moses combination because *all* of our project's target segments were filled with translations and our work essentially involved editing. This workflow consisted of the following steps:

- Segments contained TM matches were edited as required
- Segments contained MT translations were quickly evaluated for accuracy and were either erased or edited
- When an MT translation was rejected, there were two possible options: if a fuzzy match was available from the TM, it was inserted in

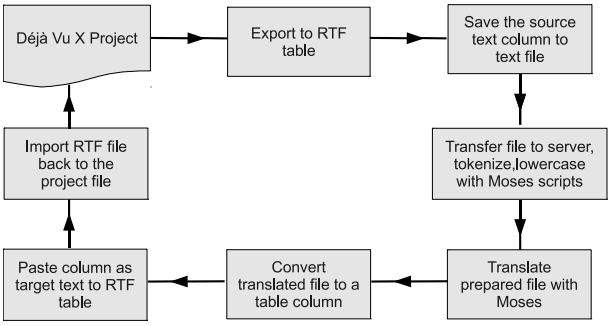

Figure 5. Steps for translating a Déjà Vu X project with Moses

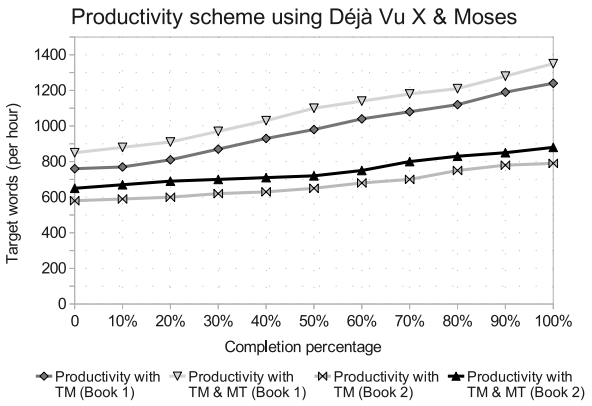

Figure 6. Productivity results with the Déjà Vu X and Moses combination

the target segment and edited; otherwise, the application's "Assemble" feature was applied

- If the assembled sentence was not of acceptable quality, it was also rejected and the translation was typed from scratch

It should be noted that a 25% of the assembled translations were accepted. This figure could have been even higher if the terminology database contained more entries (see Section 4). Our productivity with this workflow is illustrated in Figure 6. It is worth noting that the productivity increase observed with MT was lower than in the segment-by-segment workflows. The reason for that was the limited control allowed to the user, and it is further discussed in Section 3.1.

3 Results and evaluation

The results obtained from each of our workflows highlight the same fact: the application of MT into our translation projects increased our productivity to a level that would not be feasible with a typical TM workflow. However, variations in our productivity schemes do exist so the results can be divided and analyzed into relevant categories. Figure 7 shows the average productivity increase we gained with the application of MT into our workflows.

3.1 Using MT in individual segments and in the whole project

The application of MT in real time, segment by segment, seems to be more efficient and better controlled. In both real-time MT workflows (Moses and Systran), we could make quickest decisions as to whether we should keep a fuzzy match, use MT, or type the translation from scratch. The decision speed is a major issue not only for MT results but for TM matches as well, so all available resources should be presented to the user seamlessly without distracting them.

Moreover, the real-time MT approach was the only possible way to use Systran effectively. Since the entries inserted into Systran's custom user dictionary had to be coded properly, using the current segment for trial and error efforts was very efficient. Of course, this approach resulted in decreased productivity but only for the first 20% of our work. After that point, our productivity was constantly increasing, far beyond our normal levels. It should also be noted that the decreased productivity phase could be restricted or even eliminated if custom dictionaries have already been built from projects in the same domain. For our statistical MT system, however, this is not an issue because it can't be improved in real-time and thus we enjoyed increased productivity from the very start of our work. The results obtained with the Systran and Wordfast combination were very close to the ones presented by Kanavos and Kartsaklis (2008).

On the other hand, applying MT to the full project beforehand is the only possible way to use MT with CAT programs that do not offer an option for connecting to external applications. Even with this approach, however, productivity was still higher than normal, although lower than in the real-time MT methods.

3.2 Using statistical MT and rule-based MT

Both workflows increased our productivity noticeably despite their differences. The Moses workflow didn't impose any penalty in our productivity and was very efficient right from the start. With Systran, however, our productivity was initially decreased, at least until the custom dictionary was filled with adequate entries.

Although the efficiency of both workflows seems rather close, a few notes should be mentioned. Firstly, we assume all the CAT programs used in this case-study provide equal efficiency and performance. In fact, variations do exist but usually they

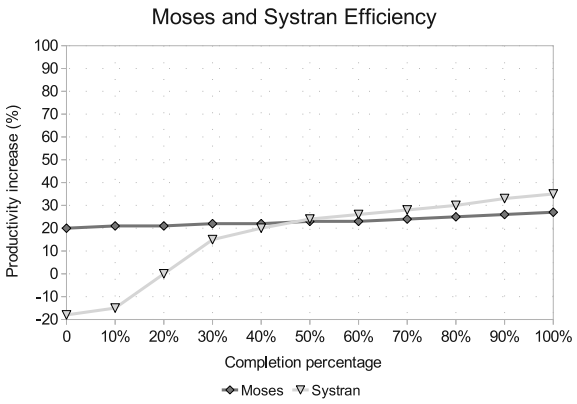

Figure 7. MT overall efficiency measurement in both real-time and non-real-time settings

are negligible for measurements in this scale, assuming the user is equally skilled in all the CAT programs. Secondly, the Systran version that was available for our target language was rather limited compared to other versions, thus the results could be better with an improved version. We should also mention that most recent Systran versions offer hybrid MT capabilities that could have yielded much better results if we had the opportunity to use them. On the other hand, the Moses workflow could have also been more efficient had we used a factored model (Koehn and Hoang, 2007). Moses is a very flexible system that can be enriched with morphological and syntactical information for more accurate decoding.

4 Discussion and assessment

The results obtained with each workflow can vary in some degree depending on a number of language, text context and style, and application-specific factors. For instance, the quantity and quality of the TM entries affect directly the performance of the Moses MT system. The more entries a TM contains, the more efficient the MT system will be since the training corpus will also be larger. Furthermore, the quality of the TM affects both the matches obtained from it and the efficiency of the MT system, for obvious reasons. Equally important is the number and the quality of the terminology entries for yielding better results from the "assemble" feature offered by the CAT programs.

Another very important factor is the domain of the translation material used to train a statistical MT system. Our workflow with Moses was successful because the TM entries with which we created the MT system had been derived from informatics books translations. This can be a serious restriction for translators who work frequently with small texts covering several different domains. In such cases, there probably won't be enough entries from each domain to build different MT systems. On the other hand, translation agencies that translate very high volumes of texts in different domains can easily build separate MT models and use the appropriate one for each specific project. Systran, however, does not have these restrictions. What matters most in a workflow with Systran is the efficiency of the rules inherent to the MT engine, which admittedly is high, and the availability of custom user dictionaries for different domains. The Systran English-Greek edition we used did not have the ability to use multiple user dictionaries, although this feature is offered in other editions and language pairs. Thus, while for a successful Moses system huge amounts of domain-specific corpora are required, Systran's custom dictionaries require a few hundred carefully coded entries.

The language pair is also a very important factor when building statistical models for Moses and coding efficient Systran user dictionaries. Greek is a morphologically and syntactically rich language that can challenge any MT system. Furthermore, it is not an ideal language for accurate "assembled" translations that some CAT programs can generate because the terminology entries used in such operations do not make use of morphological or syntactical information.

Productivity gains may also vary depending on the style of the translation work. For instance, using these workflows for translating texts with repetitive and step-by-step sections is more productive than translating texts with a more natural or academic style. However, in either case there are still benefits in productivity. It is worth noting that in all relevant measurements and comparisons the MT systems (even the mainstream Google Translate service which cannot be customized) outperformed the CAT programs' native "assemble" features.

Some other important but non-measurable factors that should be taken into consideration when evaluating productivity with these workflows are the

so-called "human factors", such as familiarity with all involved technologies, experience in the subject of the translation, volume of the translation project, ability to identify text patterns and predict MT results, usability issues, etc (Lagoudaki, 2008). For general acceptance and use, these technologies should be integrated into an intuitive end-user application. In the following section we attempt to provide some suggestions and outlines as to how such an application should be designed and what functions it should be able to perform.

4.1 A proposal for a unified application

An application that would incorporate both technologies (MT and TM) should mainly be a TM application. No matter how efficient an MT system is, professional translators will always rely on their TMs and terminology databases. Besides, our increased productivity resulted from the MT systems in our workflows was possible only because we used them on top of TM environments. What is actually needed is a user-friendly "wrapper" application that will be able to perform the steps in our methodologies automatically for the user.

4.1.1 Basic features and design

The basic components of the application should be the translation editor, where the actual translation work will be performed; the TM databases and the terminology databases; an editor for examining and revising the TM and terminology databases; and a document aligner for creating translation memories from existing translations. The translation editor should offer rich text editing features and the ability to filter and sort segments based on certain criteria. The editor's environment should also provide panes for displaying TM matches, terminology matches and, for our purpose, MT results. Selective application of MT and presenting MT suggestions as another source of reference is a key for successful integration, as it is also mentioned by Lagoudaki (2008).

Other useful features that could be implemented include concordance searches against the TM and terminology databases; ability to assemble translations from all available databases; interoperability with other CAT applications via open standards (TMX, XLIFF, TBX); automatic quality assurance control (spell-checking, number and tag checking); and robust filters for importing and exporting various file formats. These features are particularly favored by TM users (Lagoudaki, 2006).

4.1.2 MT integration

A statistical MT system (such as Moses) integrated into a CAT program can be trained with existing TMs. TM databases are usually organized into subjects so our proposed application could monitor the number of entries in domain-specific TM databases and provide the user with an option to train an MT system when this number reaches a satisfactory level (for example, 50,000 translation units). However, there are some critical restrictions with this approach, such as the computing resources and the time required for training. A solution to this problem could be the deployment of the MT component into a separate server machine, either on the user-side or in a third party's premises. This third-party could act as a service provider, so in that case the MT feature could be offered as Software as a Service (SaaS). Also, the application may offer an option to use Moses when there are no TM matches, similar to the feature offered by Wordfast Classic, and on demand by the user.

Another statistical MT-related feature that could be implemented is the ability to build factored models using the terminology databases. Most modern CAT tools allow the user to enter linguistic information to terminology entries, although this information is only used for reference purposes. Our application, however, could possibly use it in conjunction with a POS (Part-of-Speech) tagger to create factored models for better MT performance.

The statistical MT technology can also find other uses beyond its main purpose of translating text automatically, as introduced by the TransType project (Langlais et al., 2000; Esteban et al., 2004). A typical example of such use is Caitra (Koehn and Haddow, 2009), a CAT tool developed by the University of Edinburgh and based on the Moses decoder. Caitra uses Interactive Machine Translation, a technology that offers suggestions of short phrases as the translator is typing. The suggestions are offered in an intuitive way without distracting the translator who has the option to accept the suggestions or ignore them and continue typing. While the trans-

lator continues to type, the suggestions offered by the system change dynamically so the user has always the option to accept or ignore them. Trados is a well-known CAT program that implements a similar feature called "AutoSuggest".

Integrating rule-based MT features into a CAT program seems easier and more flexible. For instance, Systran requires average computing resources and does not involve any special preparation steps. However, the efficiency of Systran is greatly dependent on the quality of the custom dictionaries. Thus, our unified application should provide a way to build Systran's user dictionaries from within the application. This can be achieved via the terminology databases which may include additional fields for coding Systran's user dictionaries.

5 Conclusions

All workflows presented in this work lead to the conclusion that machine translation, either statistical or rule-based, has already matured and can increase substantially the productivity of professional translators. Although a comparison with other relevant work would have been of particular interest, to the best of our knowledge this is the first large-scale work testing multiple TM-MT configurations in real-world scenarios. Furthermore, the fact that our approach is successful with a target language as highly inflective as Greek means that we can safely expect productivity increase in many other settings involving Western language pairs. A crucial factor is the availability of in-domain corpora, as well as the domain itself. The benefits of integrating TM and MT technologies would be even greater for language service providers who work with high volumes of technical texts. However, restrictions do exist, particularly in the implementation of MT in professional workflows. Currently, there is not a straightforward way for using MT in TM environments or a translation software that integrates tightly both technologies, but work towards this purpose is in progress by the authors.

The results presented in this work show that translation workflows are likely to change in the near future and machine translation will certainly has its role in it.

References

Carl, Mark and Silvia Hansen. 1999. Linking Translation Memories with Example-Based Machine Translation. In *Machine Translation Summit VII*, pages 617–624, Singapore.

Esteban, José, José Lorenzo, Antonio S. Valderrábanos, and Guy Lapalme. 2004. Transtype2–An Innovative Computer-Assisted Translation System. In *Proceedings of the ACL-2004 Interactive Posters/Demonstrations Session*, pages 94–97, Morristown, NJ, USA.

Kanavos, Panagiotis and Dimitrios Kartsaklis. 2008. Translation and Computers (in Greek). *Scientific American,* Greek edition, 6(2):24–31.

Koehn, Philipp and Barry Haddow. 2009. Interactive Assistance to Human Translators Using Statistical Machine Translation Methods. In *Machine Translation Summit XII*, pages 73–80, Ottawa, Canada.

Koehn, Philipp and Hieu Hoang. 2007. Factored Translation Models. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL '07)*, pages 868–876, Prague, Czech Republic.

Koehn, Philipp, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open Source Toolkit for Statistical Machine Translation. In *Proceedings of the Demo and Poster Sessions of the 45th Annual Meeting of the Association for Computational Linguistics (ACL '07)*, pages 177–180, Prague, Czech Republic.

Lagoudaki, Elina. 2006. Translation Memory Systems: Enlightening Users Perspective. Key Findings of the TM Survey 2006 Carried out During July and August 2006. *Imperial College London*.

Lagoudaki, Elina. 2008. The Value of Machine Translation for the Professional Translator. In *Proceedings of the 8th Conference of the Association for Machine Translation in the Americas*, pages 262–269, Waikiki, Hawaii.

Langlais, Philippe, George Foster, and Guy Lapalme. 2000. Transtype: A Computer-Aided Translation Typing System. In *EmbedMT '00: ANLP-NAACL 2000 Workshop: Embedded Machine Translation Systems*, pages 46–51, Morristown, NJ, USA.