Getting the best out of a mixed bag

Terence Lewis
MITI

ABSTRACT

This paper discusses the development and implementation of an approach to the combination of Rule Based Machine Translation, Statistical Machine Translation and Translation Memory tecnologies. The machine translation system itself draws upon translation memories and both syntactically and statistically generated phrase tables, unresolved sentences being fed to a Rules Engine. The output of the process is a TMX file containing a varying mixture of TM-generated and MT-generated sentences. The author has designed this workflow using his own language engineering tools written in Java.

1. Introduction

There is broad agreement today that improvements in the fluency of machine translation output can be achieved by the use of approaches that harness human translations. This paper discusses the development and implementation of an approach to the combination of Rule Based Machine Translation, Statistical Machine Translation and Translation Memory technologies. This "multi-faceted approach" can be applied in a vendor and platform independent environment.

Early methods for the combination of machine translation and translation memory tools involved the use of an analysis made by translation memory software to produce an export file containing unknown segments which were then fed into a machine translation system. The results were subsequently imported into the translation memory software where they received an "MT" penalty. This technique has been superseded in practice by the introduction of MT plug-ins which are now available in the major commercial translation memory applications. Some professional translators use these plug-ins, in a "pre-translate" stage in preference to accepting fuzzy matches to produce draft translations which they then revise. In the automated translation work flows referred to above, the process is controlled through the translation memory application, and in applications like SDL Studio 2014 and memoQ the user can set which, if any, machine translation services are to be consulted.

The author has implemented an approach whereby the machine translation system itself is able to consult and draw upon translation memories and a statistical translation model as part of an automated translation process. This paper does not claim to describe a novel approach as the

literature contains many useful accounts of attempts to combine translation memory and machine translation, such as the paper by Koen & Senellart[1], and the very detailed account by Kanavos and Kartsaklis[2]of attempts to combine a variety of third-party translation tools on real translation projects. It describes a practical way of harnessing a number of different data resources which the author has found useful for handling major projects such as the one described further on in this paper. The author acknowledges that the methods discussed in this paper have been broadly applied in the recently launched MateCat project[3].

The author is both a translator and a self-taught software developer – not a computational linguist – and this contribution is intended to be a personal account of experiences rather than a scientific paper. The approach to translation automation described is a practical one. For nearly two decades the author has worked as an independent provider of translation automation services and a language technology consultant, for Siemens Nederland and other Dutch companies and institutions. The translation memories utilised to deliver these services have been built up in decades of work as a translator and language service provider. They include sentences translated by other professional translators and post-edited machine translations. All these data have been combined into one large TMX file, which is a record of the author's knowledge and experience. These services are supplied within a private network and are not delivered "in the Cloud", something that is important to many of the author's industrial and government customers.

At TC21, the author reported on his attempts to combine the use of his growing translation memories with his early Dutch-English Machine Translation system.[4] His clients in those days mostly wanted to receive fully formatted MS Word files. His initial efforts relied on the use of the Trados Translator's Workbench to create export files of unknown words which were machine-translated, with the translations being imported into the translation memory. Nowadays, however, corporate clients for his automatic translation services have themselves acquired commercial translation environments – commonly but not exclusively SDL Studio and memoQ, and want to receive the output of the MT system in a format such as TMX which can be imported directly into their translation memory software for post-editing. Significantly, the supply of a TMX file is billed at a lower rate than a fully formatted MS Word file!

Having worked extensively as a translator for companies in the chemical, transportation and telecommunication industries, the author has built up a wide-ranging "master translation memory". As a developer of a machine translation application he has investigated various ways of exploiting these translation memories on the assumption that output derived from human translations will generally be more fit for purpose than that generated solely by the application of syntactic rules.

[1]Convergence of Translation Memory and Statistical Machine Translation, P.Koen & J. Senellart, MT Marathon 2010.

[2]Integrating Machine Translation with Translation Memory: A Practical Approach, P. Kanavos & D. Kartsaklis, JEC 2010.

[3]See www.matecat.com

[4]The Best of Both Worlds – or will two mongrels ever make a pedigree? , T. Lewis, TC21, 1999.

2. Background

The approach described in this paper was a practical response to the challenges posed by a large translation automation project. The author was asked to translate 250,000 words in hundreds of small files for the HAN University of Applied Sciences[5] in the Netherlands. The translations were needed quickly and costs had to be kept down. After comparing the results of various MT services offering Dutch-English translation, the university decided to avail itself of the author's machine translation software. At the time of placing the order, the university had not even decided exactly how the machine translation output was going to be processed further. Working with veteran Localization Consultant, Lou Cremers, the author decided on a workflow which involved machine translating a TMX file in such a way that the client would receive a raw translation memory. The HAN translation office wanted to post-edit this machine translation output in a translation memory environment so that it could be brought to a standard fit for publication on the university's website. The institution eventually decided to use memoQ and arranged for a post-editing team to be trained in its use. After terminological preparation of the project, the MT software produced a series of TMX files (TMX 1.1). The post-editors were given an opportunity to provide feedback which was incorporated into the translation memory being built as the project progressed and even led to the improvement of some of the MT rules.

When a second project for the HAN came along, the author knew that many of the sentences in this project already had translations in the translation memory built from the first project. The client wanted to handle the project in the same way as the first one and receive a TMX file. Wishing to carry out the new project completely via his own machine translation software rather than use third-party translation memory software, the author wrote the code to enable his machine translation engine to search this translation memory directly in order to enjoy the benefit of the translations contained in that memory within the automatic translation process.

3. The process

In terms of file handling, the process is a simple one. A TMX file is prepared[6] in which the target elements, in this case **<tuv lang="EN-GB">,** initially contain the source text. The MT engine reads the input TMX file line by line. Only the content of the target element is of interest, everything else being written straight to the output buffer. The engine first sends off a query to the TranslationMemoryConsulter class. If the selected translation memory contains a 99% or 100% match, the corresponding translation is entered in the target element in place of the "source text". After making any required minor editorial adjustments to the target segment, the engine moves on to the next segment. If the selected translation memory does not contain a suitable match, the segment is sent off to the internal translation server for further processing. The translated content is returned from the server segment by segment and also replaces the source text in the target element.

The output of the process is a TMX file containing a mixture of directly TM-generated and MT-generated sentences. The client's translators can review this file for "sanity checking", post-editing or full-blown revision, depending on the intended purpose of the automatic translation.

[5]See http://blog.han.nl/onlineeducation/vertalen-van-teksten-en-site-diverse-tools/
[6]This can be done using the Okapi Tools: http://www.opentag.com/okapi

The advantage of delivering a TMX file is that the post-editing work can be done in any commercial (or non-commercial) translation environment, in a dedicated TMX editor such as Olifant or even in a simple text editor on any platform. The translators at the author's main client for language technology services – Siemens Nederland N.V. - import the MT-generated TMX file directly into their translation memory in SDL Studio 2011. Other clients use different CAT tools.

As stated above, the MT engine goes through the submitted TMX file on a segment by segment basis, and if a translation memory contains a 100% or 99% match, the target language segment is inserted in the output TMX file and the engine then moves onto the next segment in the input file. The MT engine also recognises segments entirely in English (often the case in Dutch documents) and inserts these directly into the output TMX file.

In practice, parts of sentences in the input file will frequently match the content of the translation memory database at subsegment level. In translation memory terminology, these are the "fuzzy matches" for which users can determine an acceptability percentage (many translators set this threshold at 75-80%) in a "Pretranslate" run. The problem with these "fuzzy matches" is that a sentence in the translation memory can be displayed to the user as an 80% match or higher, even though it means the exact opposite of the source sentence. Figure 1 gives a simple example of this.

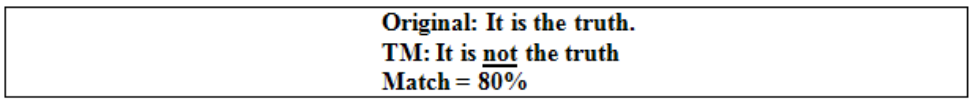

Figure 1: The problem with fuzzy matches

From the earliest translation memory environments attempts have been made to use colours or other devices to alert the user to the fact the displayed target segment is not a complete translation of the source segment. However, the author found that users of his Dutch-English automated translation service, who were paying for "unrevised machine translation", were not prepared to receive translations potentially containing glaring inaccuracies. For this reason, it was decided to set a very high threshold (99-100%) for transferring segments automatically from the translation memory into the output file via the MT engine. On the other hand, his translation memories contained millions of potentially useful segments.

This awareness of possessing translation memories that didn't always tell the whole truth has led the author to investigate ways of storing potential subsegment matches in phrase tables which the machine translation engine can consult. The advantage of storing data at subsegment level is that the translations retrieved by the MT engine are **NOT** fuzzy matches but 100% matches for the part of the sentence to which they correspond. In practice, the author's MT engine consults two phrase tables to search for matches at subsegment level: one is created by the application of syntactic rules; the other is statistically derived.

The **Clean Data Repository** is created by decomposing segments in translation memories into meaningful fragments or subsegments. The term "clean" refers to the fact that the data have been checked by a human reviewer. They have been assembled by automatically aligning

meaningful subsegments of the segments contained in translation memories. This is done by "looping through" source and target sentences from complete sentence down to bigram level. Through the application of a series of syntactic rules, source and target segments are divided into noun phrases, prepositional phrases and verbal phrases, and short sentences are also retained as sentences. Many of these entries will correspond to Multiword Expressions[7] (MWE's). The human checking goes beyond grammar checks; it is made sure that terms are in-domain and the subsegment is in the right register. Entries in the Clean Data Repository in the author's Dutch-English machine translation system look like these:

```
<trans-unit>
<source xml:lang="nl-NL">bij het uitvoeren van een beveiligingsfunctie</source>
<target xml:lang="en-GB"><mf>when a security function is performed</mf></target>
</trans-unit>
 <trans-unit>
<source xml:lang="nl-NL">virtueel diagnostisch systeem</source>
<target xml:lang="en-GB"><n1>Virtual Diagnostic System</n1></target>
</trans-unit>
<trans-unit>
<source xml:lang="nl-NL">wenst over te gaan tot</source>
<target xml:lang="en-GB"><vts>wishes to proceed to</vts></target>
</trans-unit>
```

Figure 2: Examples of entries in Clean Data repository

These data are stored on-disk in the form of an XLIFF file and are loaded into a Java data structure at run-time. No translation scores are involved as it is assumed that any entry in the Clean Data Repository, having been checked, is 100% correct, or has a probability of 1. The user can add project-specific data to this repository on the fly before a translation run. This is done by breaking down the source document into ngrams, which are presented with their frequency of occurrence as shown in Figure 3. The user can add the translations to the source segments in a text file and the program then converts the entries into the XLIFF format and adds them to the repository.

```
1 : zijn de volgende afspraken gemaakt welke verder in het
1 : welke verder in het document uitgebreider zullen worden toegelicht
1 : in het kort zijn de volgende afspraken gemaakt
```

Figure 3: Phrases to be translated and added to Repository

The author's program also allows post-edited TMX files to be "decomposed" so that the subsegments are added to the repository on the fly – this is particularly useful on large projects with multiple files as improvements made by post-editors/revisers can be incorporated into the data repository immediately, literally by the click of a mouse button. Figure 4 shows subsegments extracted from a TMX file ready to be added to the Clean Data Repository.

[7] For a simple explanation of Multiword Expressions, see http://en.wikipedia.org/wiki/Multiword_expression

```
<trans-unit>
<source xml:lang="nl">plannen is voorbehouden aan</source>
<target xml:lang="en"><mf>planning is reserved to</mf></target>
</trans-unit>
<trans-unit>
<source xml:lang="nl">in de Excel sheet staat</source>
<target xml:lang="en"><mf>the Excel sheet contains</mf></target>
</trans-unit>
<trans-unit>
<source xml:lang="nl">de aanwezigen bij het overleg</source>
<target xml:lang="en"><n2>those attending the consultation</n2></target>
</trans-unit>
```

Figure 4: Subsegments automatically extracted from a TMX file

One of the problems of using a static phrase table is that in real language phrases are not set in stone but come to life in an engagement with other words. The Dutch verbal phrase "brengen op de hoogte", literally translated as "bring someone on the height", means "to inform". We inform somebody about something so the software has to enable us to link "brengen" to "op de hoogte" while taking into account an intervening object. The author has written code to deal with these "gappy phrases", so that a sentence like "hij bracht de raad op de hoogte" will be correctly translated as "he informed the board". A special repository of such gappy phrases is built as a subset of the Clean Repository Data, using segments contained in the main repository, which has in turn been derived from the main translation memory.

If the MT engine fails to find matches for every subsegment in the Clean Data repository it may proceed to consult a "dirty data" repository. As the name suggests, the entries in this phrase table will not have been individually reviewed. Their accuracy is reliant upon the successful building of a statistical translation model using the tools provided in the Moses Statistical Translation Toolkit[8]. The repository is built using the *train-model.perl* – script. Existing translation memories are saved as TMX files and then divided into source and target language text files, which are subsequently tokenized and cleaned in the manner known to SMT practitioners. Train-model.perl is employed in the same way as for building the statistical translation model to be used by the Moses decoder in a statistical machine translation environment. The resulting phrase tables will be as general or specific as the TMX files from which they are derived.

The content of the file phrase-table.gz generated by the "Moses" training process is stored as byte code in a serialised Java data structure, or HashMap. The translation model is created is created during a training stage, i.e. in advance of its deployment. The data to be stored in this Java data structure are selected during the training operation on the basis of scores produced by train-model.perl, which means that the entire phrase table created by train-model.perl will not be stored in memory at run-time. At present, the generated scores are utilised in a fairly naive way - low-scoring phrase pairs are excluded from the phrase table loaded into memory. There is certainly scope for a more sophisticated use of these scores.

At run-time, the MT engine "decides", sentence by sentence, on the basis of a number of criteria whether the Statistical Model should be used to translate that particular sentence. These

[8] http://www.statmt.org/moses/

criteria include the length of the sentence, the complexity of the sentence and the percentage of subsegments that have already been translated with subsegments from the Clean Data Repository. For example, the Dutch sentence “Het product wordt getest” (The product is tested) will not be sent to the SMT model because the Rules Engine can perfectly well handle it, and it is probably also in the Clean Data Repository. One the other hand, in the case of the more complex sentence “In het ideale geval wordt er niet getest” (In the ideal situation no testing is done) a search will be made in the statistically derived phrase table. This is consulted, not by the open source Moses decoder, but by the author's own decoder.

As already stated, the Clean Data Repository mostly comprises noun phrases and verb phrases and short whole sentences. Unlike the results of the translation memory search, the target subsegments retrieved from the Clean Data Repository are POS tagged and may even contain some semantic information. The “clean data” are therefore useful input to the Rules Engine which resolves any untranslated parts of the sentence and composes the subsegments from these different sources into the final English sentence. On the other hand, the “dirty” or unvetted, statistically derived data do not currently include any syntactic or semantic information. Their relevance and usefulness is dependent upon the domain relevance of the TMX files from which they are derived. Figure 5 shows a translation unit derived from the Statistical Machine Translation model. In most translation memory applications, the source of a translation is displayed in the editing panel so it is easy for the post-editor to identify whether the translation comes from a translation memory, via a Statistical Translation Model or from the Rules Engine.

```
<tu creationdate="2014/10/15 09:41" creationid="SMT">
<tuv lang="NL-NL">
<seg>Actielijsten moeten worden aangevuld cq worden beheerd.</seg>
</tuv>
<tuv lang="EN-GB">
<seg>Action lists must be supplemented or, as the case may be, managed. </seg>
</tuv>
</tu>
```

Figure 5: Translation unit supplied via the Statistical Machine Translation model

On a large project it is possible to gauge the usefulness of particular resources by running test files and then consulting the job log. Figure 6 shows a job log from a project on which both a translation memory and a statistical machine translation model supplied translations. The Clean Data Repository is consulted by default and its supplied translations are always piped into the Rules Engine.

```
Starting new logfile
Number of errors logged: 0
Segments translated from Translation Memory= 35
Segments/subsegments supplied by Clean Data Repository= 195
Segments translated by Rule-Based MT Engine= 661
Segments produced from SMT Data= 45
```

Figure 6: Example of a job logfile showing translation sources

Any segments (or subsegments) not resolved by the above three approaches will then be tackled by the Rules Engine, which has its own ways of dealing with multiword expressions. The whole process – from input to output – is summarised in Figure 7.

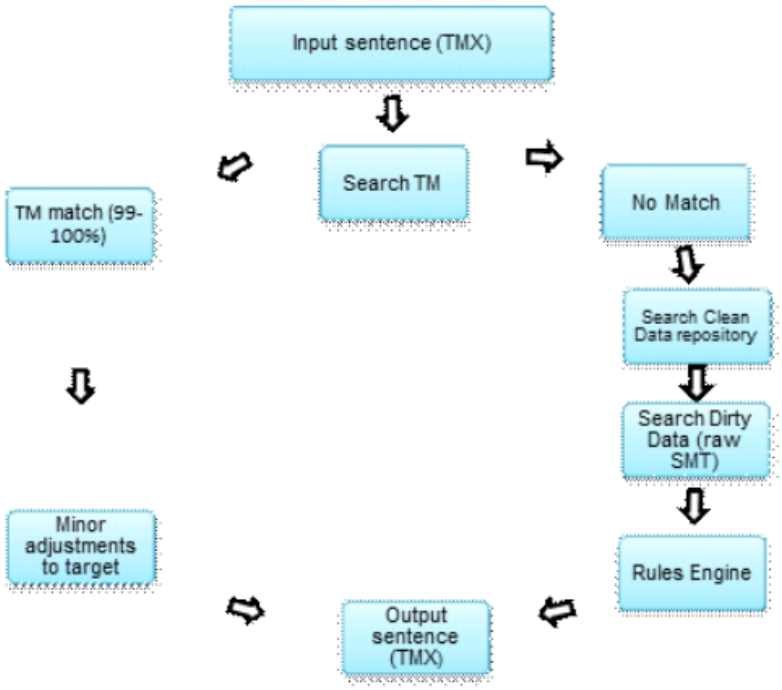

Figure 7: Automatic translation process from input to output

4. Practical experience

This three-pronged approach is useful for large-scale translation automation projects in domains for which a relevant translation memory is available. Typically, a project memory, which is a subset of the main memory, is created. That translation memory is then used to generate new entries in the Clean Data Repository. A statistical translation model may then be trained from the same translation memory. It has to be remembered that the resulting statistically derived phrase table is not used as the sole supplier of the translation of the sentence but only to offer translations of phrases at subsegment level.

In the "HAN project" described above, the preparatory work and the feedback of the translation team was the key to its success, which led the university to consider the adoption of this form of automated translation for future projects. The translators had an opportunity to provide translations of a system-generated list of unknown words and phrases in advance of the first translation run. The unigrams (single words) were entered in the Core Dictionary, while the phrases were added to the Clean Data Repository. The project involved three translation runs of a document of some 250,000 words. After run_1 and run_2 respectively the data were corrected as some of the target sentences were considered unacceptable for submission to the post-editors. The author fed the corrections into the system by adjusting entries in the translation memory and entering phrases and complete sentences into the Clean Data repository. The results achieved in run_3 were judged acceptable as they met the basic undertaking that the sentences would be grammatically correct and the output would not contain any garbage or

jumbled phrases. At the end of the project, the translation team was handed a large translation memory.

5. Judgement on usefulness

The SMT Research Survey Wiki[9] lists more than 60 publications (since 2005) dealing with domain restriction. The literature suggests that the restriction of the domain in which training is done delivers results that require less correction of technical terms in post-editing than general-purpose resources. The author's practical experience of how different types of output from his software are received by his customers bear out this claim.

In day-to-day practice the author implements domain restriction by using a project translation memory and entering relevant subsegments in the Clean Data Repository. This has broadly been found to deliver a more fluent – though sometimes disjointed - output than use of the Rules Engine alone. The disjointedness is frequently one of style, reflecting the multiplicity of translation originators, and the disconnect is between sentences rather than within a sentence. In particular, it has been found that long-standing translation memory entries that have never been updated are sometimes less accurate in terms of terminology than the "pure MT" output.

This methodology has been used to provide English translations of Dutch technical texts to members of international teams on large-scale projects in the fields of transport, highway engineering and healthcare. In nearly two decades of providing machine translation services, the author has only received two serious complaints about the quality of the output. The key to user acceptance has undoubtedly been the Clean Data Repository, which is continually enlarged and kept up to date by deconstructing "approved" translation memories, something which is done at the end of every project. In fact, many of its short sentences, such as technical instructions, could equally have been included in a translation memory, and a script allows them to be converted from the XLIFF to the TMX format so that they can be used in any third-party translation memory such as Studio 2014.

The incorporation of unsupervised or "dirty data" is recent and the extent to which the use of the statistically derived model improves the output of the MT run has not yet been fully investigated. Figure 8 shows the English output from the machine translation of a Dutch sentence. The first version is generated by piping the results from the Clean Data Repository straight into the Rules Engine. The second version is built partly by finding matches in the statistically derived phrase table.

***DUTCH*: De uitvoer van het rapport dient geëxporteerd te kunnen worden (als excel-, csv- en pdf-file).**

***RBMT*: It must be possible to export the output of the report (as Excel, csv and pdf-file).**

***SMT MODEL*:**
The output with the report should be capable of being exported (such as excel, csv and pdf-file).

Figure 8: Output from Rules Engine and from Statistical Translation model

[9] http://www.statmt.org/survey/Topic/DomainAdaptation

In the above example the Rules Engine achieves a more fluent output than that produced if the SMT model is accessed (Google Translate provides a similar translation to our in-system SMT model). The conversion of the awkward passive "should be capable of being exported" to "it must be possible to export" is the result of the application of a specific rule. However, the translation provided by the SMT model is not POS-tagged so the Rules Engine has less to work on. The author is therefore considering POS-tagging of the English phrases translated from the SMT model "on the fly". This POS-tagging can be accomplished by the BNC_Frequency_List_Investigator, a Java class written by the author to retrieve the parts of speech of items contained in the British National Corpus.

Nevertheless, at subsegment level, the author's general-purpose Statistical Model (trained from Europarl[10]) can provide reasonably credible translations as shown below[11]:

Dutch	**With SMT**	**Without SMT**
beschikbare personeel	human resources available	available personnel
gewenste middelen	desired equipment	desired means
onderdeel van de planning	component of the timetable	component of the planning
ingreep	surgical procedure	intervention
bijzondere aandacht voor	particular attention to	particular attention for
er kan ingesteld worden	you can set	it can be set
bezetting van de OK	workforce of the OR	occupation of the OR
medische gegevens	medical statistics	medical data
wordt er gekeken naar	we look at	it is looked to
in de volgende functies	within the next functions	in the following functions
actuele omstandigheden	topical situations	current circumstances
registratie van het OK-team	registration of surgical team	record of the surgical team
indien een patiënt overlijdt	where a patient dies	if a patient dies
standby functie	standby task	standby function

The above phrases are taken from a healthcare specification which was translated both with and without the data provided by the SMT model. Given that the SMT model was trained from

[10] http://www.statmt.org/europarl/

[11] Phrases extracted from machine translation of healthcare technology specification – with and without use of SMT

the broad-ranging Europarl data without any tuning, it seems reasonable to assume that models trained on in-domain data will yield at least comparable results in terms of the usability of subsegments. The author does not intend to throw out his Rules Engine but rather to use the SMT model as an additional source of translations at subsegment level.

6. Conclusions

This flexible approach to automatic translation has been designed for handling large-scale projects involving professional translators at the beginning and end of the production line. The author prepares these projects in close collaboration with the translators, who will ultimately import the "machine output" into their respective translation memory tools. The full process sees the machine translation engine first consulting any provided translation memory and then, if there is no TM match, consulting a Clean Data Repository of subsegment data and (if the engine so decides) a statistically derived phrase table, before piping the "mixed bag" of phrases into a Rules Engine which generates the final English sentence.

Based on his results and customer acceptance achieved from using a large general-purpose translation memory and a statistical translation model based on the Europarl corpus, the author plans to build a series of in-domain translation memories and use them to train in-domain statistical translation models.